Vorlesung Algorithmische Kartografie

Flächenkartogramme

LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

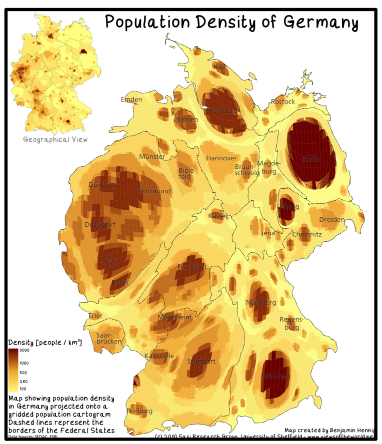
Martin Nöllenburg

02.07.2013

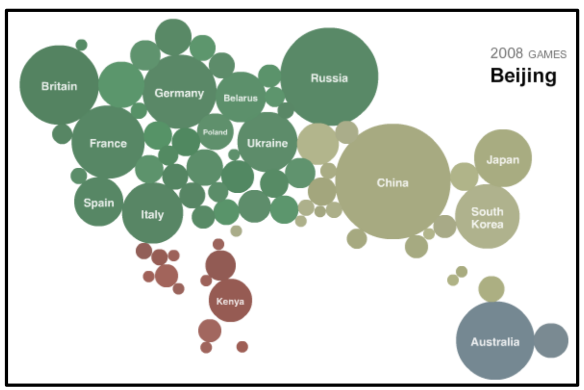
Flächenkartogramme

Def.: Ein **Flächenkartogramm** (dt. *Kartenanamorphote*) ist eine Kartendarstellung, in der jede Flächeneinheit proportional zu einer externen Größe und nicht mehr zur tatsächlichen Fläche ist (z.B. Bevölkerungszahl).

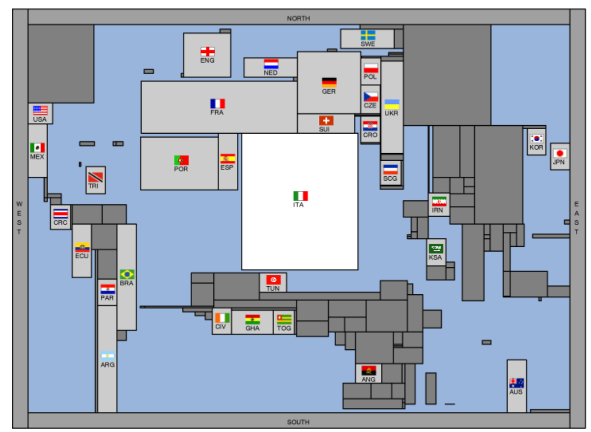
→ Form, Lage und Nachbarschaften der Regionen werden verzerrt

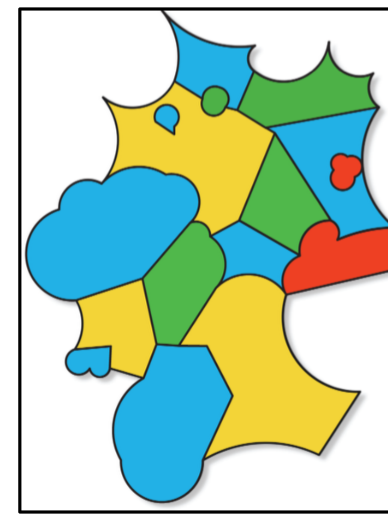


©Benjamin Hennig



©New York Times


©Bettina Speckmann



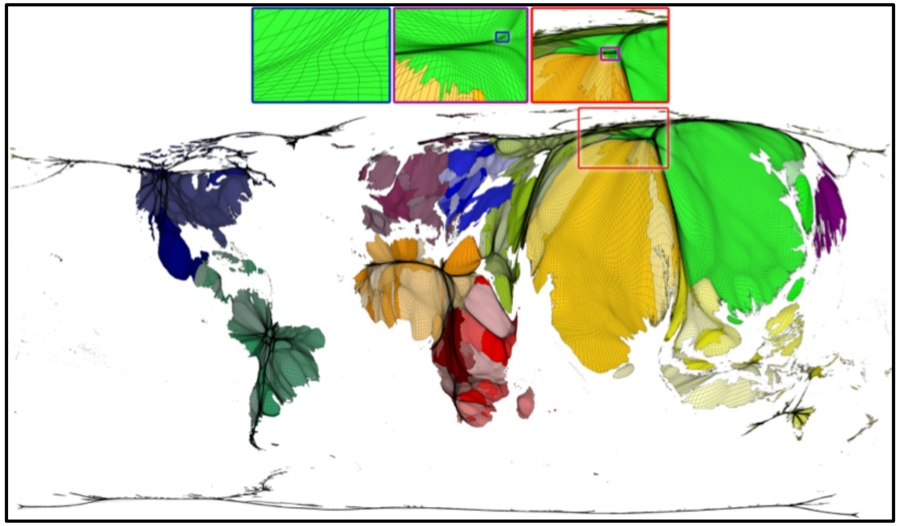
Qualitätskriterien:

- Wiedererkennbarkeit der Form
- Flächenvergleichbarkeit
- Lage der Regionen
- korrekte Adjazenzen
- kleiner Flächenfehler
- geringe Komplexität
- Ablesen der Fläche

Zusammenfassung Diffusionskartogramme

Diskussion:

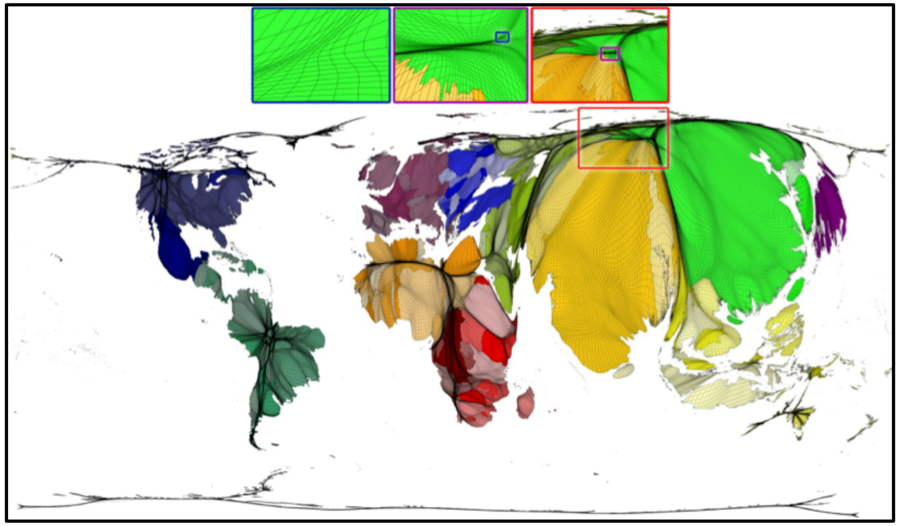
- Wiedererkennbarkeit der Form ○
- Flächenvergleichbarkeit ⊖
- Lage der Regionen ⊕ ○
- korrekte Adjazenzen ⊕
- kleiner Flächenfehler ⊕ ○
- geringe Komplexität ⊖
- Ablesen der Fläche ⊖



Zusammenfassung Diffusionskartogramme

Diskussion:

- Wiedererkennbarkeit der Form ○
- Flächenvergleichbarkeit ⊖
- Lage der Regionen ⊕ ○
- korrekte Adjazenzen ⊕
- kleiner Flächenfehler ⊕ ○
- geringe Komplexität ⊖
- Ablesen der Fläche ⊖

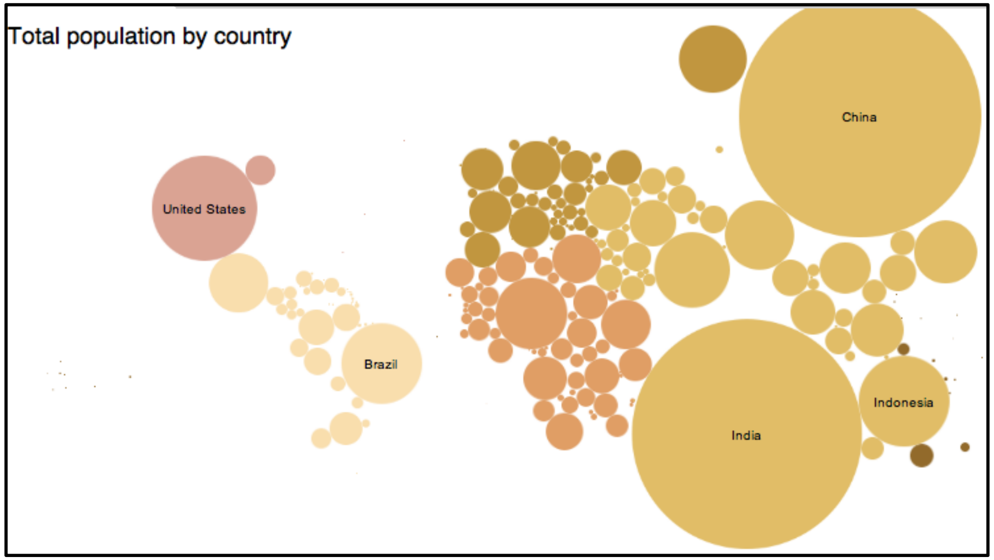


Film!

Zusammenfassung Kreiskartogramme

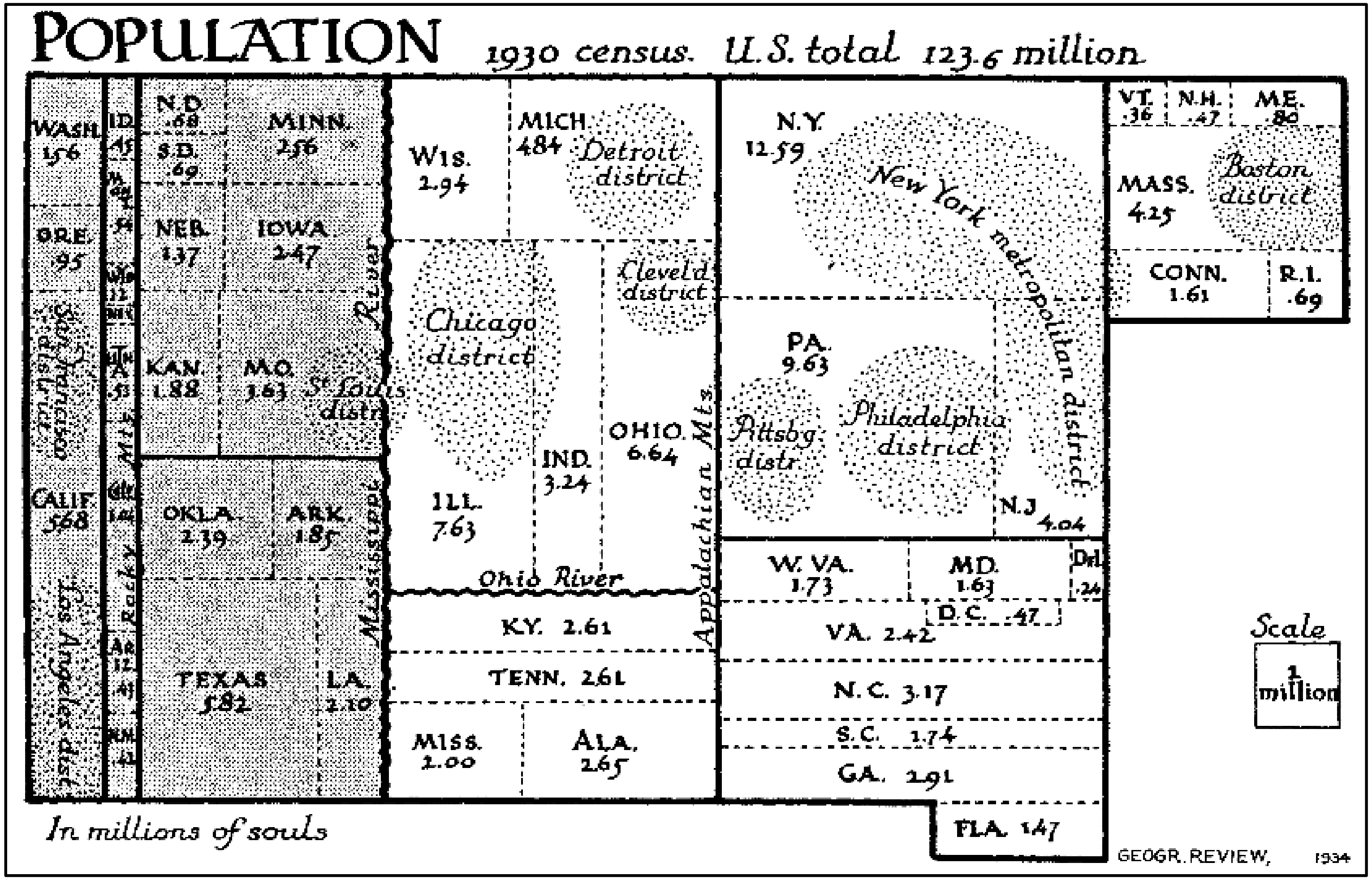
Diskussion:

- Wiedererkennbarkeit der Form ⊖
- Flächenvergleichbarkeit ⊕
- Lage der Regionen ⊖ ○
- korrekte Adjazenzen ⊖ ○
- kleiner Flächenfehler ⊕
- geringe Komplexität ⊕
- Ablesen der Fläche ⊕ ○



Rechteckskartogramme

- jede Region als Rechteck repräsentiert
- gegebene Zielflächen
- trade-off korrekte Flächen/korrekte Adjazenzen



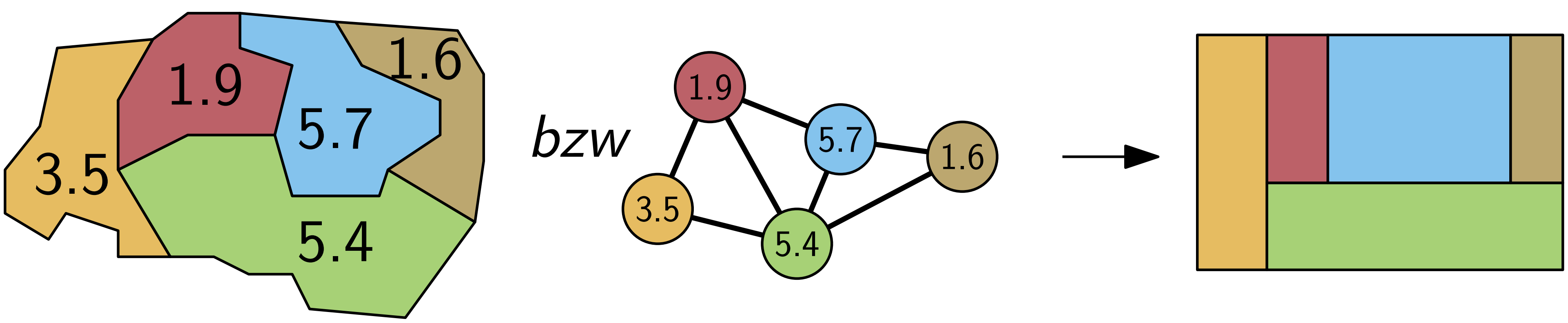
Problemstellung

Geg: politische Karte M (Rechtecksunterteilung), positives Gewicht w_i für jede Region R_i

bzw: knotengewichteter intern triangulierter planar eingeb. Graph G dual zu M, Knoten v_i entspricht Region R_i, Kanten zw. adjazenten Regionen, Knotengewichte w_i

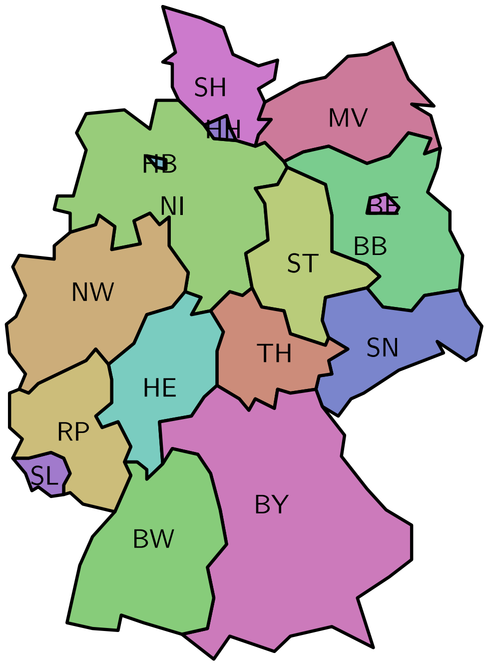
Ges: verzerrte Karte M' äquivalent zu M mit $|R_i| = w_i$

bzw: flächenproportionale Kontaktrepräsentation von G, jeder Knoten v_i als geometrisches Objekt s_i mit Fläche w_i, so dass s_i und s_j sich berühren gdw. $v_i v_j \in E$

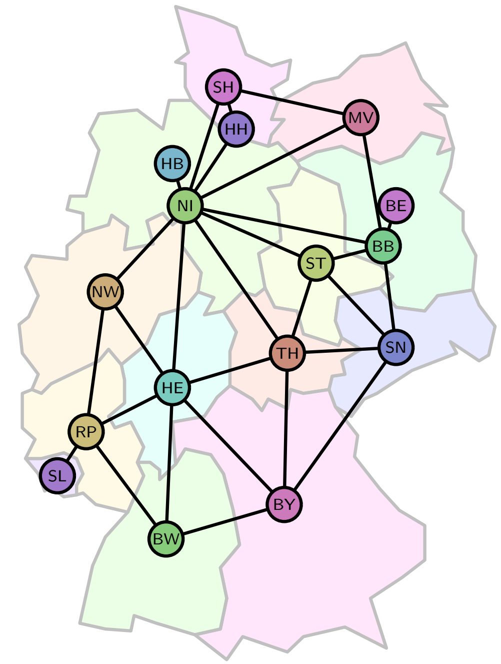


Überblick des Verfahrens [van Kreveld, Speckmann '07]

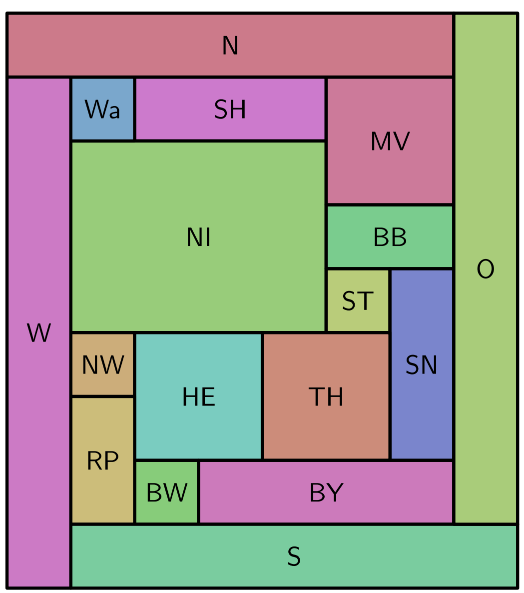
Karlsruhe Institute of Technology



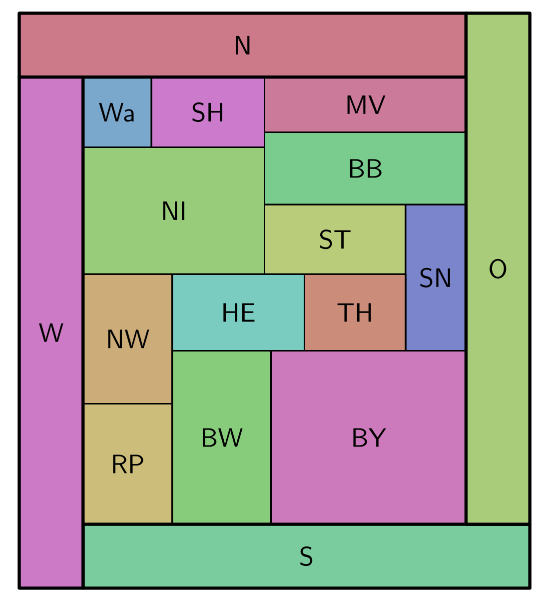
Eingabekarte



Dualgraph



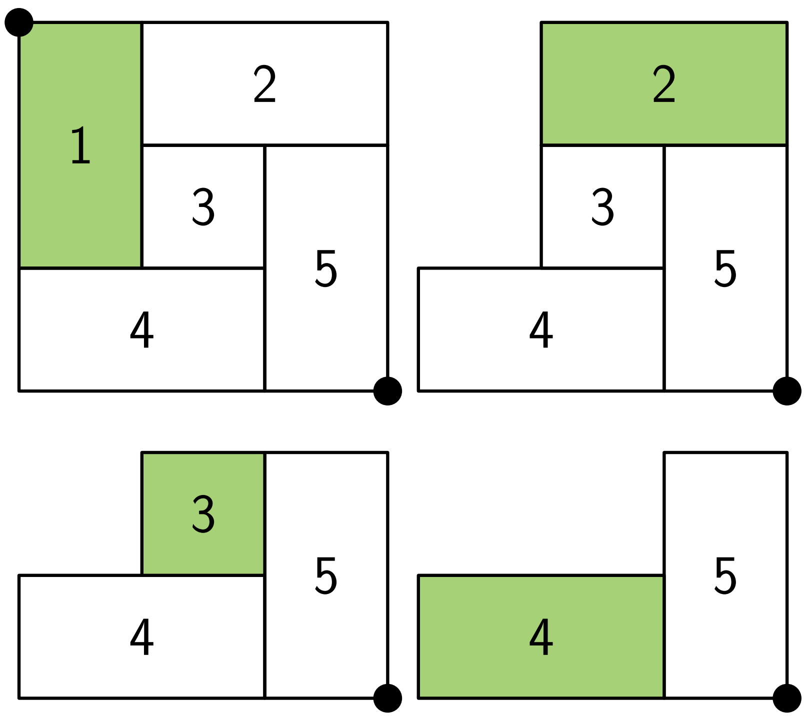
Rechtecksdual



Kartogramm

L-zerlegbare Layouts

Ein irreduzibles Rechteckslayout $\mathcal{R}$ heißt **L-zerlegbar**, falls es eine Folge $(R_1, R_2, \ldots, R_n)$ der Rechtecke von $\mathcal{R}$ gibt, so dass R_1 und R_n in gegenüberliegenden Ecken von $\mathcal{R}$ liegen und jedes Polygon $\cup_{j=i}^{n} R_j$ L-förmig ist.

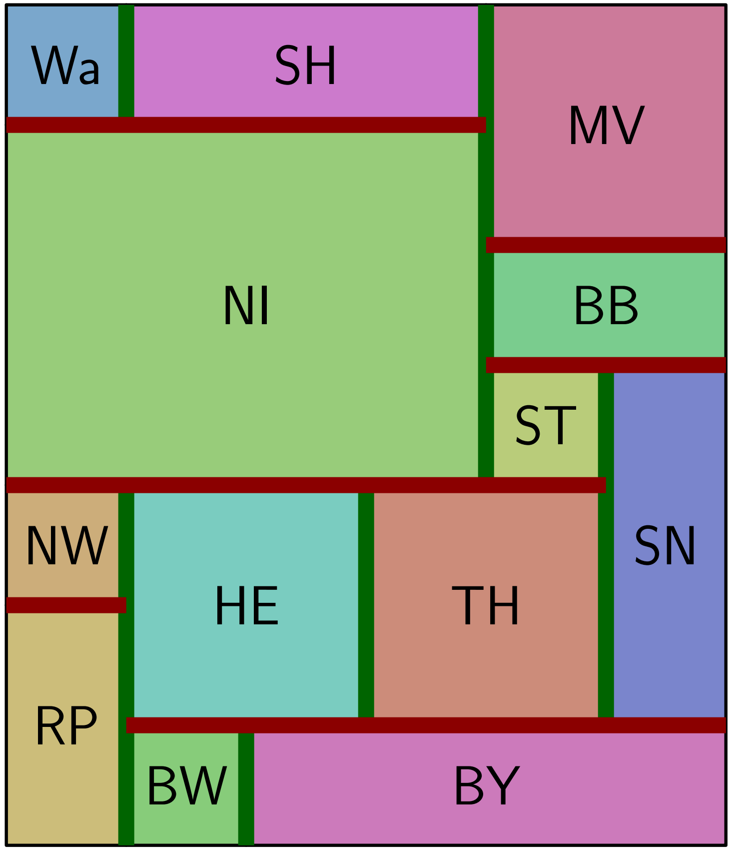


L-Zerlegungssequenz

Satz:
Ein L-zerlegbares Layout hat entweder genau eine oder keine Lösung als Kartogramm ohne Flächenfehler und mit korrekten Adjazenzen.

Heuristik für Rechteckskartogramme

Nicht jedes Rechtecksdual ist L-zerlegbar, nicht jede Flächenzuweisung für L-zerlegbare Layouts hat eine Lösung.



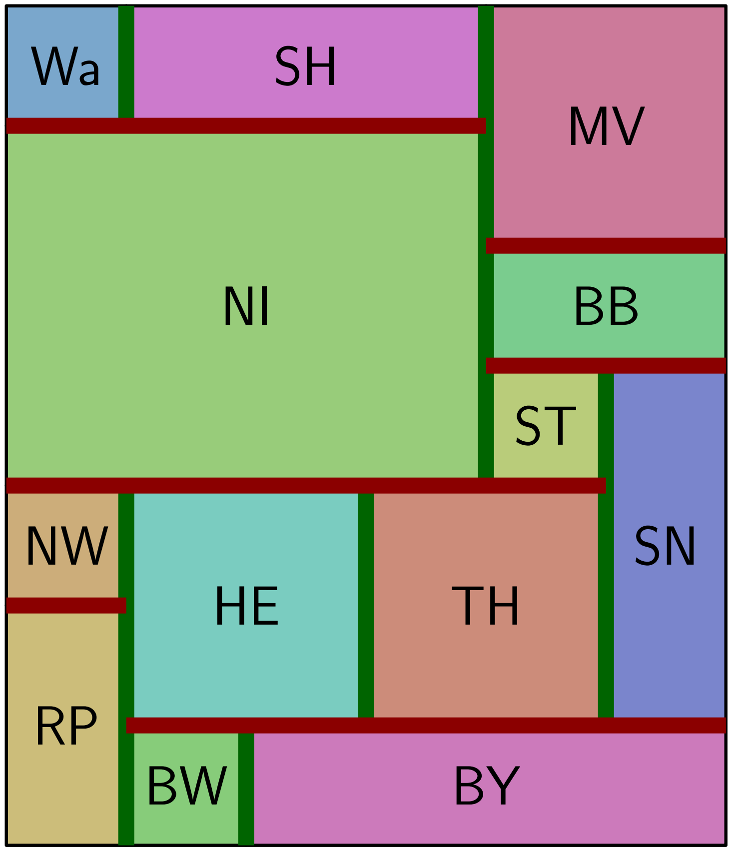
SegmentMoving

while lokale Verbesserung möglich **do**
- $s \leftarrow$ bel. maximales Segment
- bewege s in bessere Richtung
- ggf. berücksichtige Adjazenzen
- ggf. berücksichtige aspect ratio

- liefert immer ein Layout
- findet lokale Optima
- Wasser benötigt keine Zielfläche
- keinerlei Garantie oder Konvergenz bewiesen

Heuristik für Rechteckskartogramme

Nicht jedes Rechtecksdual ist L-zerlegbar, nicht jede Flächenzuweisung für L-zerlegbare Layouts hat eine Lösung.



Demo!

SegmentMoving

while lokale Verbesserung möglich **do**
- $s \leftarrow$ bel. maximales Segment
- bewege s in bessere Richtung
- ggf. berücksichtige Adjazenzen
- ggf. berücksichtige aspect ratio

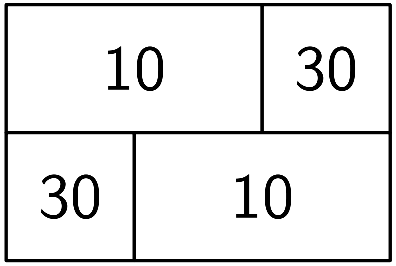
- liefert immer ein Layout
- findet lokale Optima
- Wasser benötigt keine Zielfläche
- keinerlei Garantie oder Konvergenz bewiesen

Am Rande: Flächenuniverselle Layouts

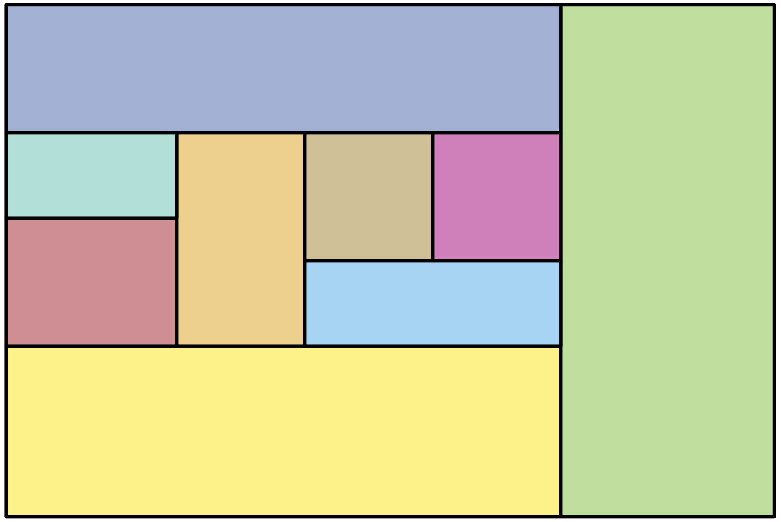
Einseitige Rechteckslayouts sind **flächenuniversell** (und umgekehrt), d.h. sie lassen sich für jede beliebige Flächenzuweisung realisieren.

[Eppstein et al. '12]

Ein Layout heißt **einseitig**, falls jedes maximale Segment auf einer Seite nur an ein einziges Rechteck angrenzt.



nicht einseitig

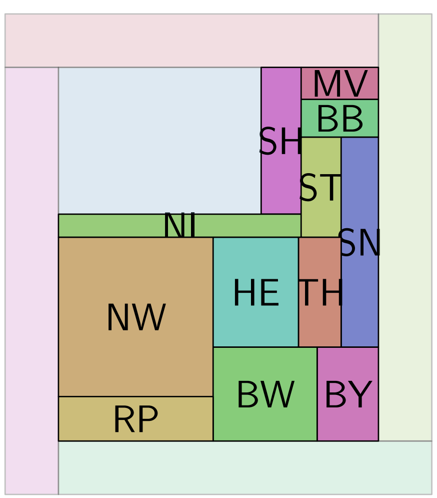


einseitig

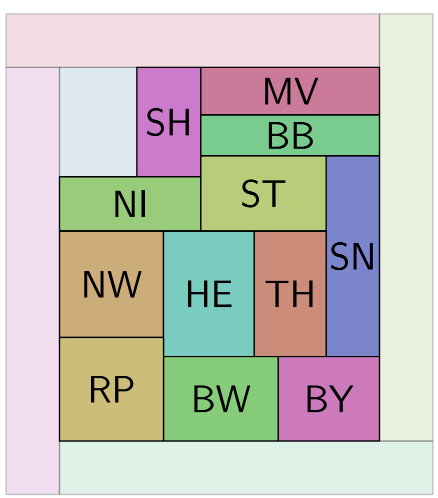
Zusammenfassung Rechteckskartogramme

Diskussion:

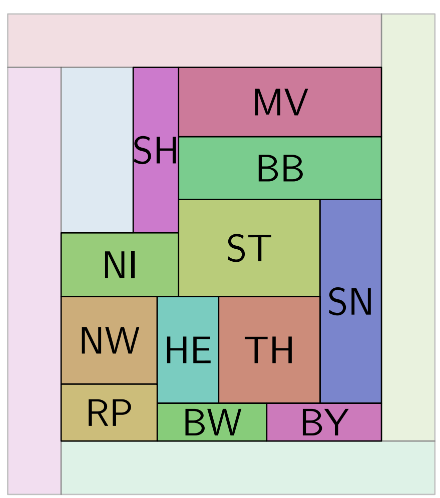
- Wiedererkennbarkeit der Form
- Flächenvergleichbarkeit
- Lage der Regionen
- korrekte Adjazenzen
- kleiner Flächenfehler
- geringe Komplexität
- Ablesen der Fläche



Bevölkerung



Studierende pro Einwohner

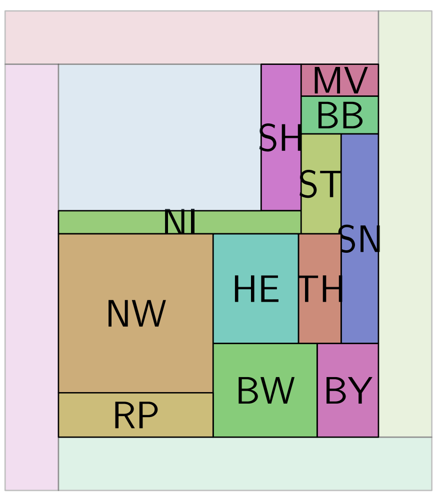


Arbeitslosenquote

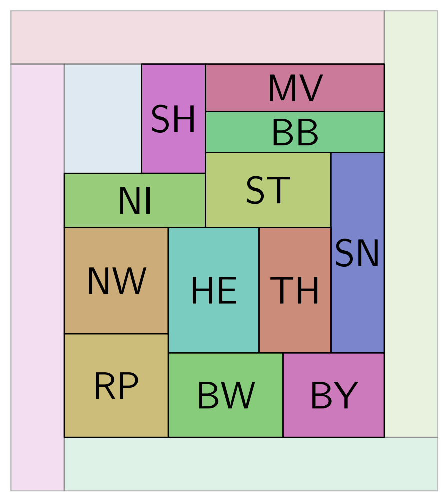
Zusammenfassung Rechteckskartogramme

Diskussion:

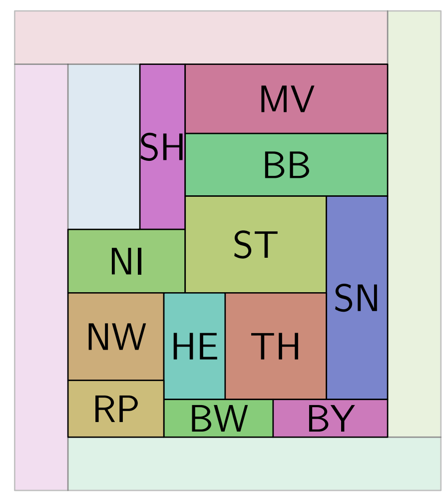
- Wiedererkennbarkeit der Form ⊖
- Flächenvergleichbarkeit ⊕ ○
- Lage der Regionen ○
- korrekte Adjazenzen ⊕/○
- kleiner Flächenfehler ○/⊕
- geringe Komplexität ⊕
- Ablesen der Fläche ○



Bevölkerung



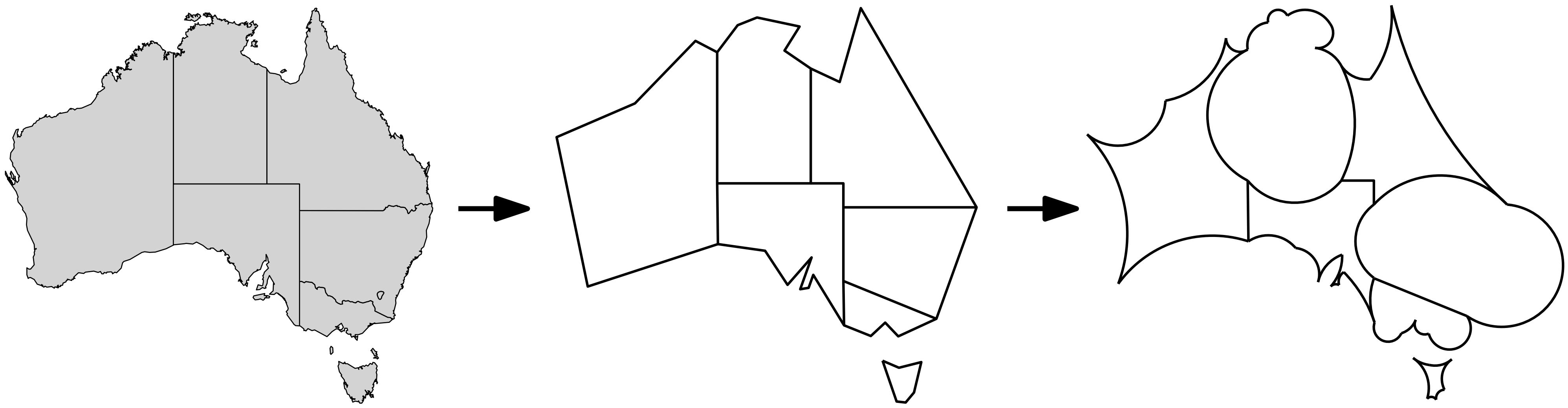
Studierende pro Einwohner



Arbeitslosenquote

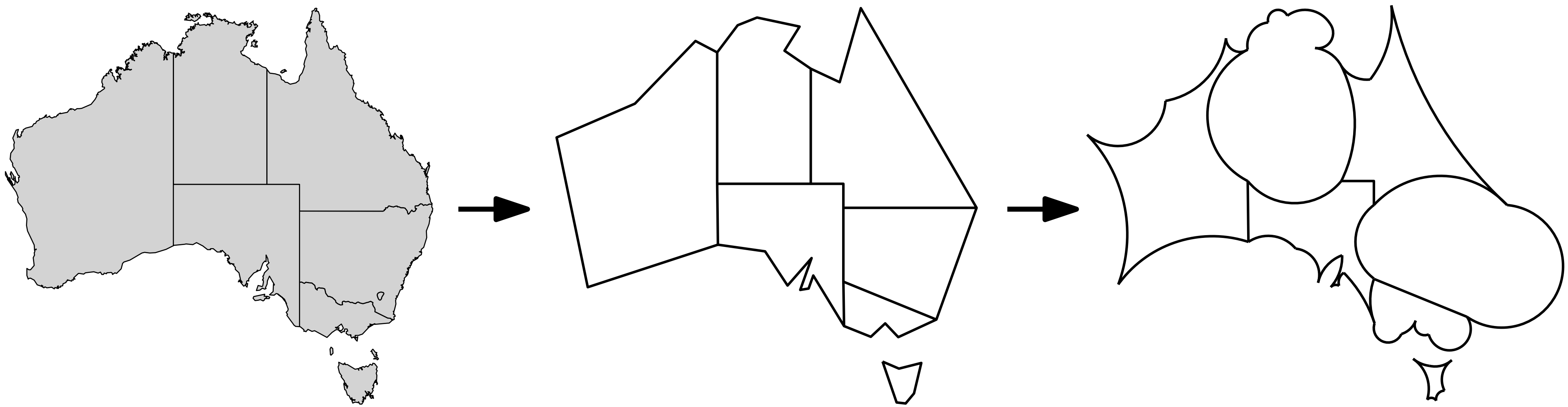
Circular-Arc Cartograms [Kämper et al. '13]

Idea:
- take region boundaries of M
- simplify them
- transform every polygon edge into a circular arc
- adjust the arc radii to obtain target areas



Circular-Arc Cartograms [Kämper et al. '13]

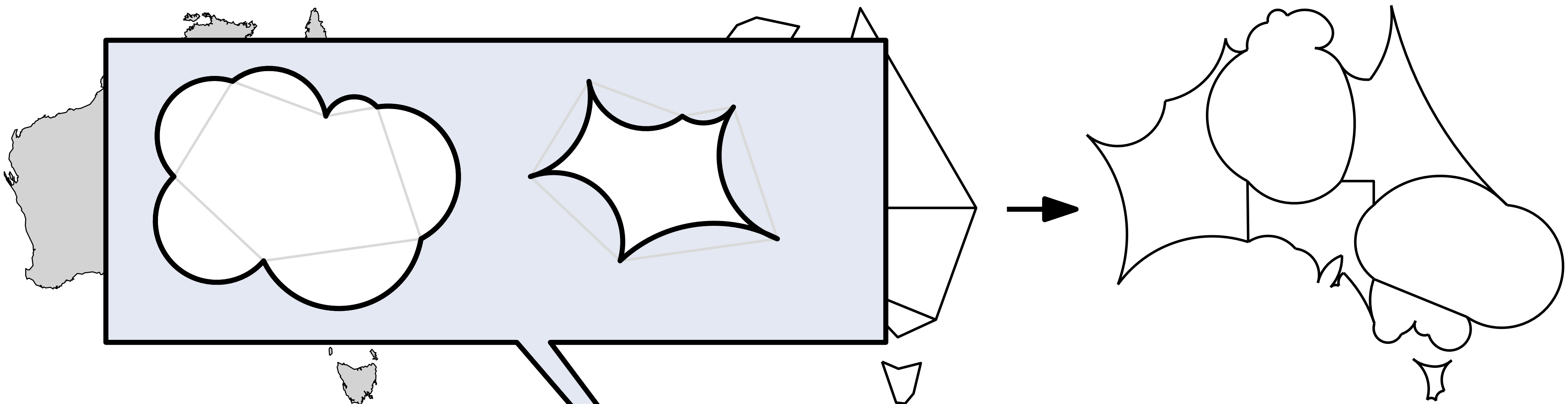
Idea:
- take region boundaries of M
- simplify them
- transform every polygon edge into a circular arc
- adjust the arc radii to obtain target areas



Properties:
- vertices keep positions $\rightarrow$ no displacement
- correct adjacencies
- similar shapes for moderate area changes
- intuitive cloud and snowflake look
- the more simplified the more area potential

Circular-Arc Cartograms [Kämper et al. '13]

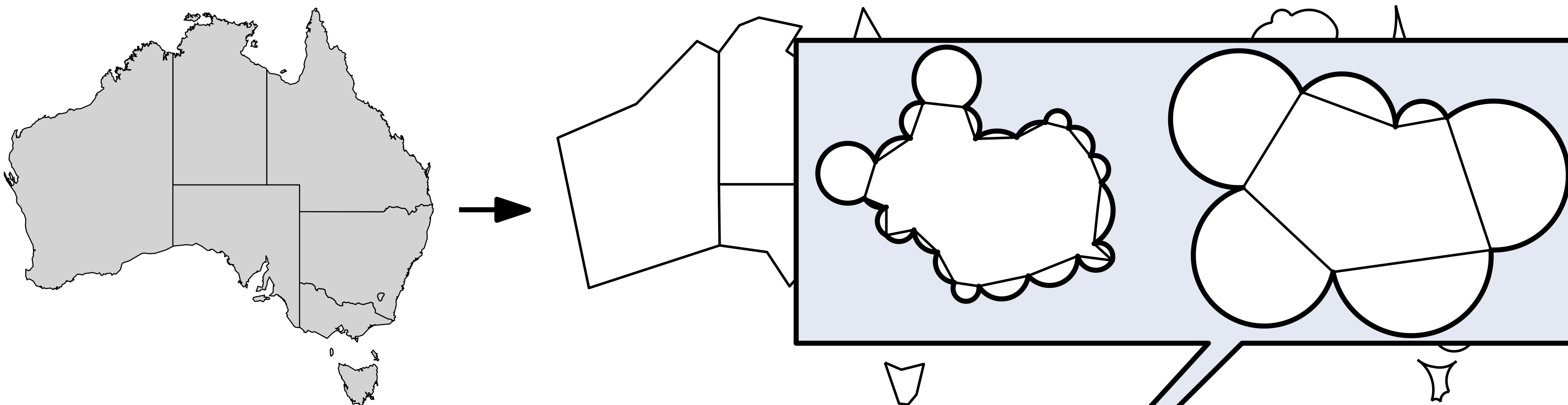
Idea:
- take region boundaries of M
- simplify them
- transform every polygon edge into a circular arc
- adjust the arc radii to obtain target areas



Properties:
- vertices keep positions $\rightarrow$ no displacement
- correct adjacencies
- similar shapes for moderate area changes
- intuitive cloud and snowflake look
- the more simplified the more area potential

Circular-Arc Cartograms [Kämper et al. '13]

Idea:
- take region boundaries of M
- simplify them
- transform every polygon edge into a circular arc
- adjust the arc radii to obtain target areas



Properties:
- vertices keep positions $\rightarrow$ no displacement
- correct adjacencies
- similar shapes for moderate area changes
- intuitive cloud and snowflake look
- the more simplified the more area potential

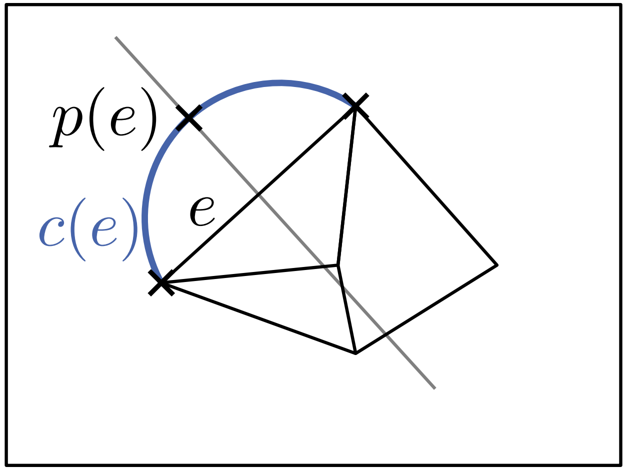
An Algorithmic Problem

Problem: Circular-Arc Cartogram (CAC)

in: (simplified) polygonal subdivision of a rectangle with faces $f_1, \ldots, f_n$ of areas $a_1, \ldots, a_n$ and target areas $t_1, \ldots, t_n$

out: a point $p(e)$ on the bisector of each polygon edge e s.t.

- e is replaced by the unique circular arc $c(e)$ through its endpoints and $p(e)$ $\rightarrow$ modified faces $f'_1, \ldots f'_n$



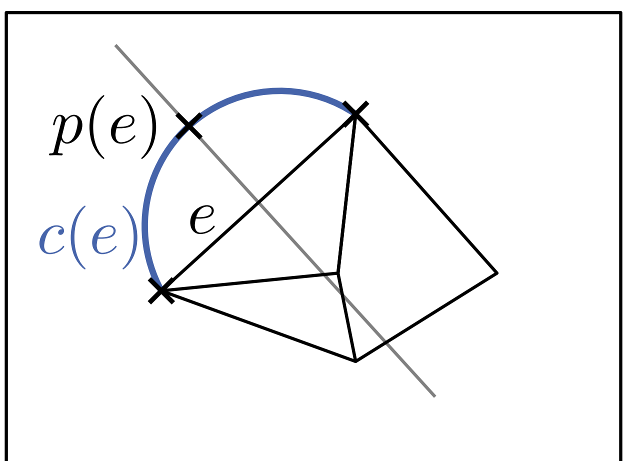
An Algorithmic Problem

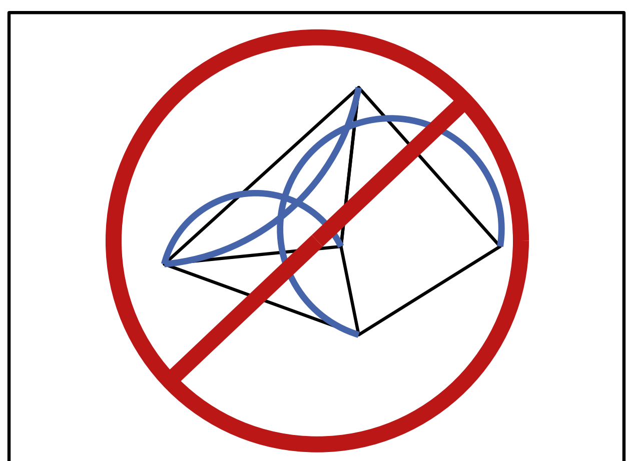
Problem: CIRCULAR-ARC CARTOGRAM (CAC)

in: (simplified) polygonal subdivision of a rectangle with faces $f_1, \ldots, f_n$ of areas $a_1, \ldots, a_n$ and target areas $t_1, \ldots, t_n$

out: a point $p(e)$ on the bisector of each polygon edge e s.t.

- e is replaced by the unique circular arc $c(e)$ through its endpoints and $p(e)$ → modified faces $f'_1, \ldots f'_n$
- no two arcs $c(e)$, $c(e')$ cross





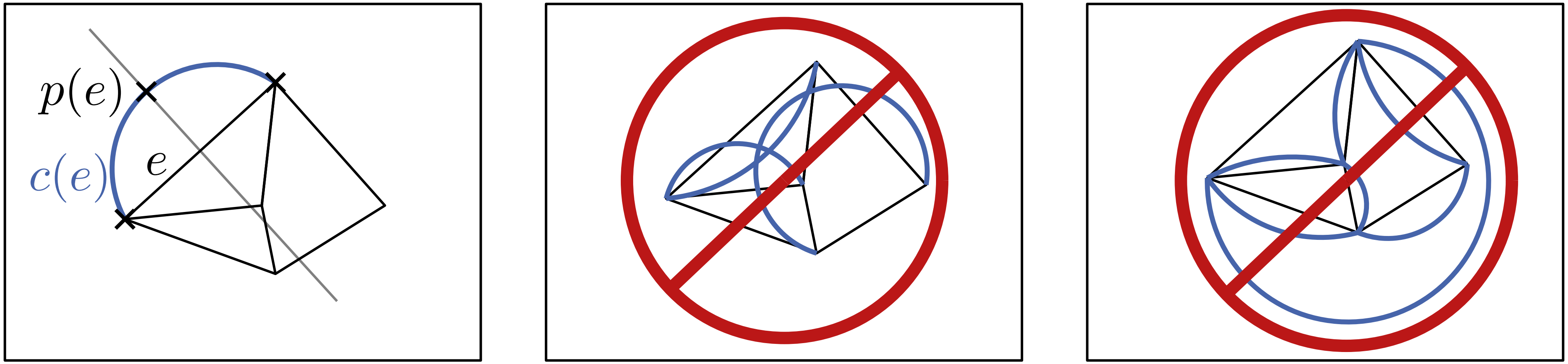
An Algorithmic Problem

Problem: CIRCULAR-ARC CARTOGRAM (CAC)

in: (simplified) polygonal subdivision of a rectangle with faces $f_1, \ldots, f_n$ of areas $a_1, \ldots, a_n$ and target areas $t_1, \ldots, t_n$

out: a point $p(e)$ on the bisector of each polygon edge e s.t.

- e is replaced by the unique circular arc $c(e)$ through its endpoints and $p(e)$ $\rightarrow$ modified faces $f'_1, \ldots f'_n$
- no two arcs $c(e)$, $c(e')$ cross
- the topology of the subdivision is preserved



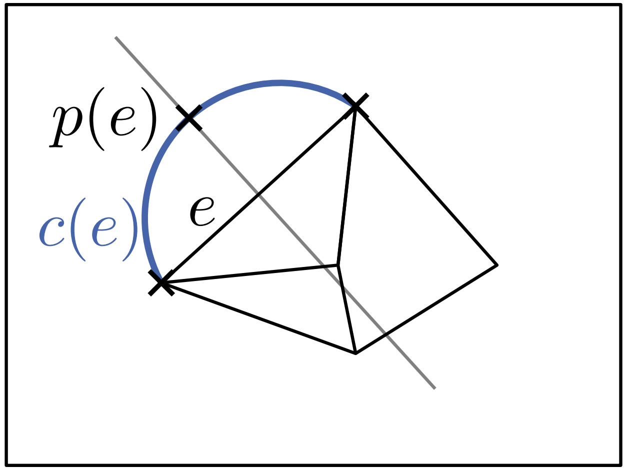
An Algorithmic Problem

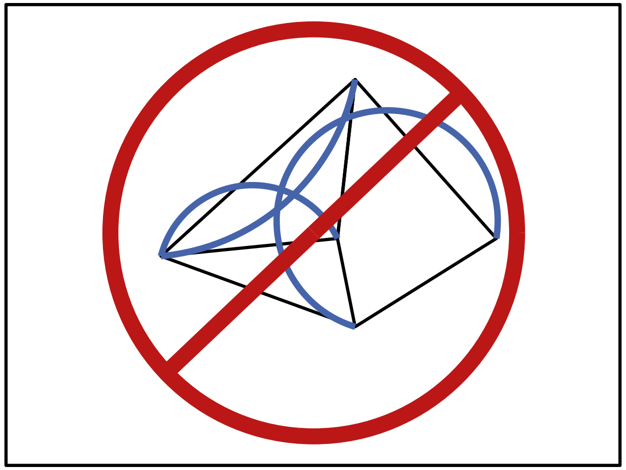
Problem: Circular-Arc Cartogram (CAC)

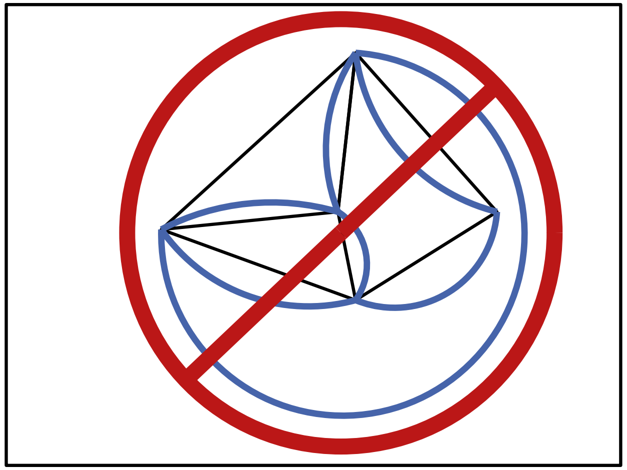
in: (simplified) polygonal subdivision of a rectangle with faces $f_1, \ldots, f_n$ of areas $a_1, \ldots, a_n$ and target areas $t_1, \ldots, t_n$

out: a point $p(e)$ on the bisector of each polygon edge e s.t.

- e is replaced by the unique circular arc $c(e)$ through its endpoints and $p(e)$ $\rightarrow$ modified faces $f'_1, \ldots f'_n$
- no two arcs $c(e)$, $c(e')$ cross
- the topology of the subdivision is preserved
- the area b_i of each face f'_i equals / is close to t_i







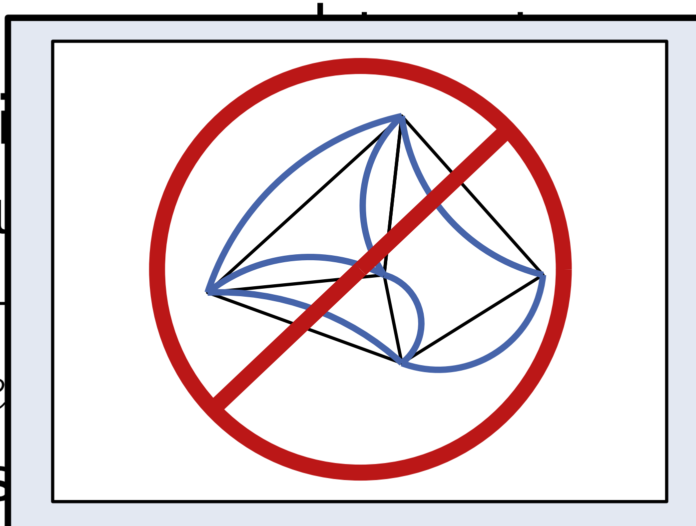
An Algorithmic Problem

Problem: Circular-Arc Cartogram (CAC)

in: (simplified) polygonal subdivision of a rectangle with faces
$f_1, \ldots, f_n$ of areas $a_1, \ldots$ s $t_1, \ldots, t_n$

out: a point $p(e)$ on the bi edge e s.t.

- e is replaced by the u through its endpoints and $p(e)$ – $\ldots f'_n$
- no two arcs $c(e)$, $c(e$
- the topology of the s
- the area b_i of each face f'_i equals / is close to t_i



Variant: strong CAC

- if $t_i - a_i \geq 0$ for a face f_i then no arc of f'_i is bent inward
- if $t_i - a_i \leq 0$ for a face f_i then no arc of f'_i is bent outward

$\rightarrow$ guarantees correct cloud and snowflake shapes

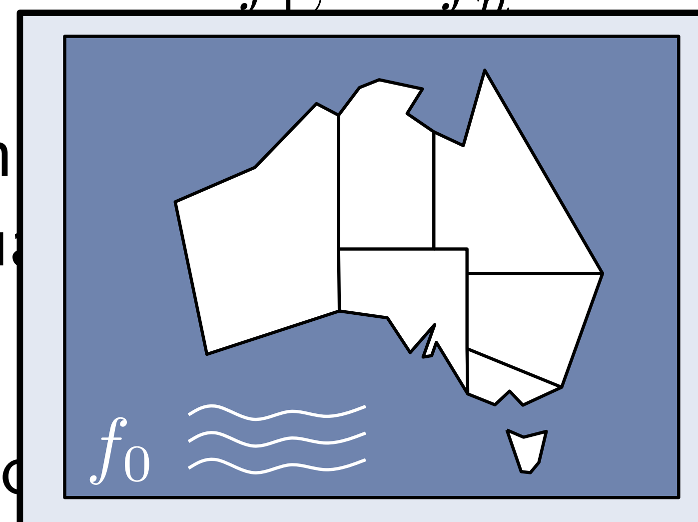
An Algorithmic Problem

Problem: CIRCULAR-ARC CARTOGRAM (CAC)

in: (simplified) polygonal subdivision of a rectangle with faces $f_1, \dots, f_n$ of areas $a_1, \dots, a_n$ and target areas $t_1, \dots, t_n$

out: a point $p(e)$ on the bisector of each polygon edge e s.t.

- e is replaced by the unique circular arc $c(e)$ through its endpoints and $p(e)$ $\rightarrow$ modified faces $f'_1, \dots f'_n$
- no two arcs $c(e)$, $c(e')$ cross
- the topology of the subdivision
- the area b_i of each face f'_i equ



Variant: strong CAC

- if $t_i - a_i \geq 0$ for a face f_i then no ard
- if $t_i - a_i \leq 0$ for a face f_i then no arc of f'_i is bent outward

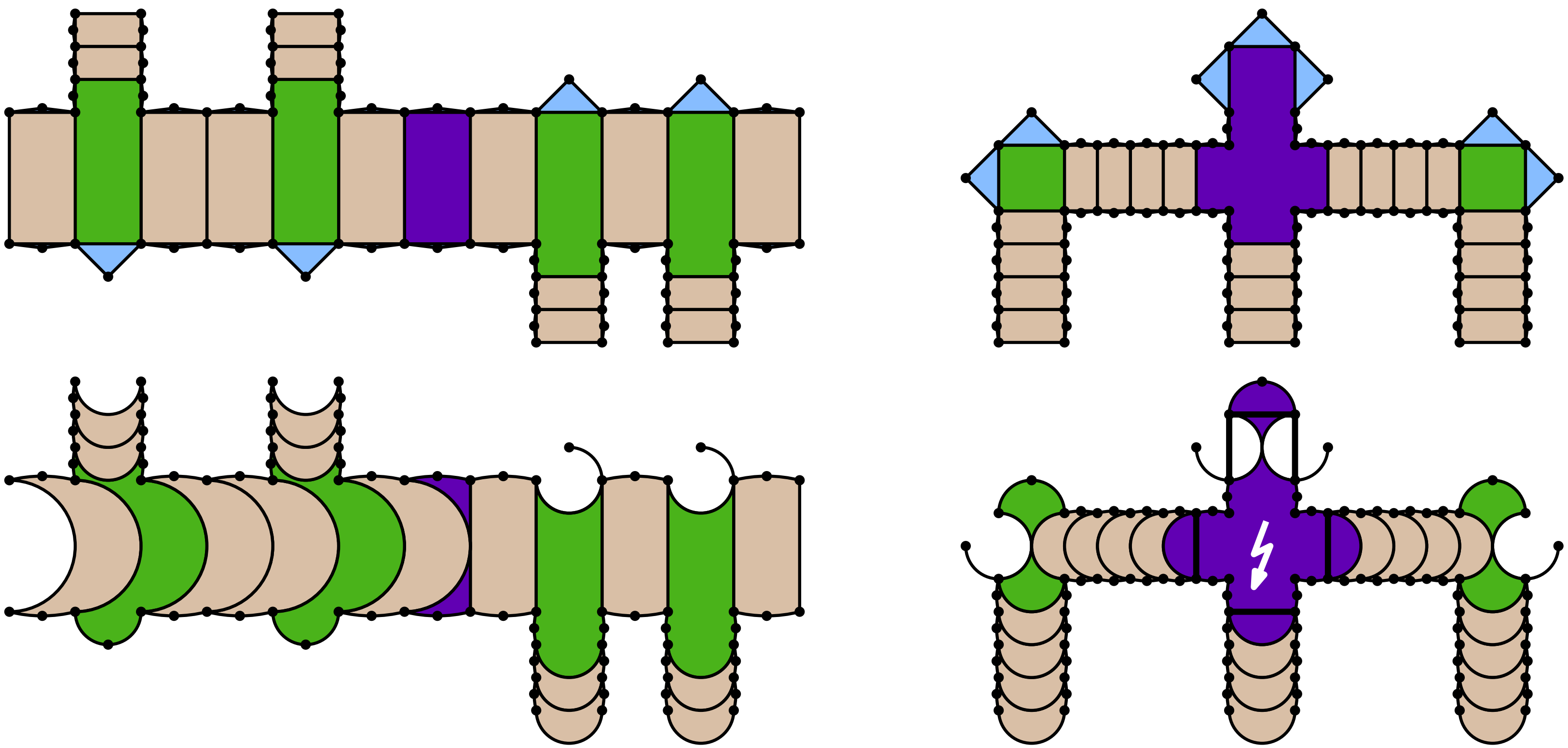
$\rightarrow$ guarantees correct cloud and snowflake shapes

Sea face: often there is a special unconstrained face f_0

Complexity

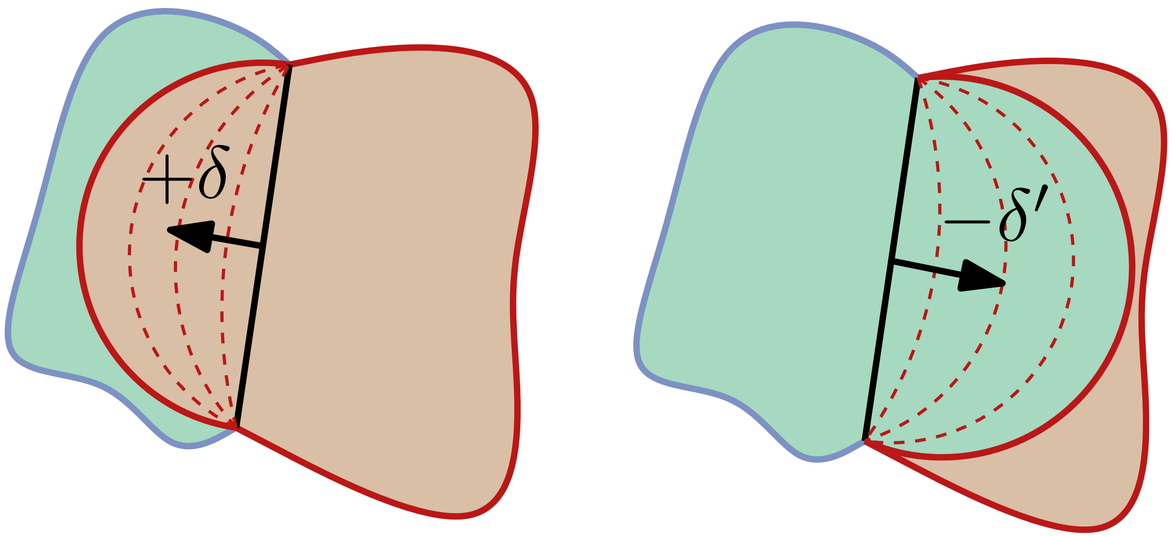
Theorem
Circular-Arc Cartogram is NP-hard, i.e., it is NP-hard to decide if an instance has an error-free circular-arc cartogram.

Proof:
Reduction from planar Monotone 3-Sat.



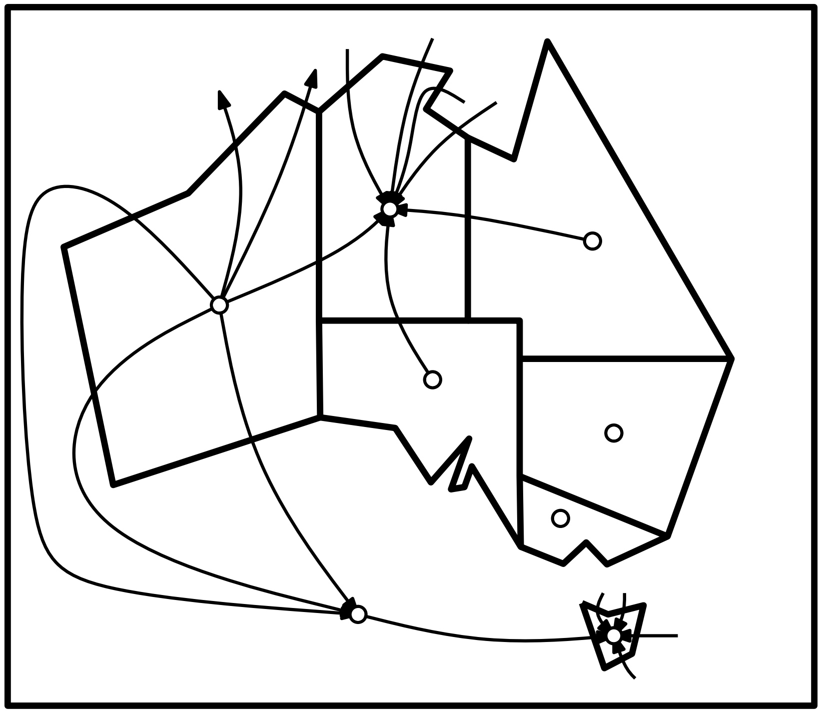
Heuristic Algorithm

Idea: bending an edge transfers area from one face to another



Heuristic Algorithm

Idea: bending an edge transfers area from one face to another
$\rightarrow$ define a **network flow model** s.t. valid maximum flow corresponds to circular-arc cartogram with small error

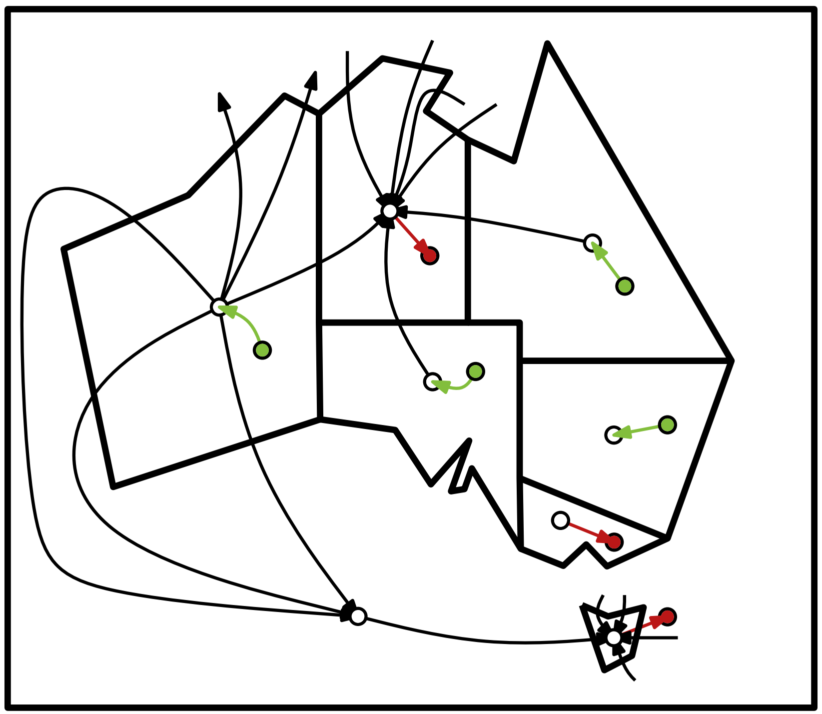


not all edges shown

- construct directed **dual graph**:
  one dual vertex v_i per face f_i,
  one dual edge per polygon edge
- **strong CAC:** direct edges outward if $|t_i - a_i| > 0$ & inward if $|t_i - a_i| < 0$
- **weak CAC:** bidirected pairs of edges

Heuristic Algorithm

Idea: bending an edge transfers area from one face to another
→ define a **network flow model** s.t. valid maximum flow corresponds to circular-arc cartogram with small error

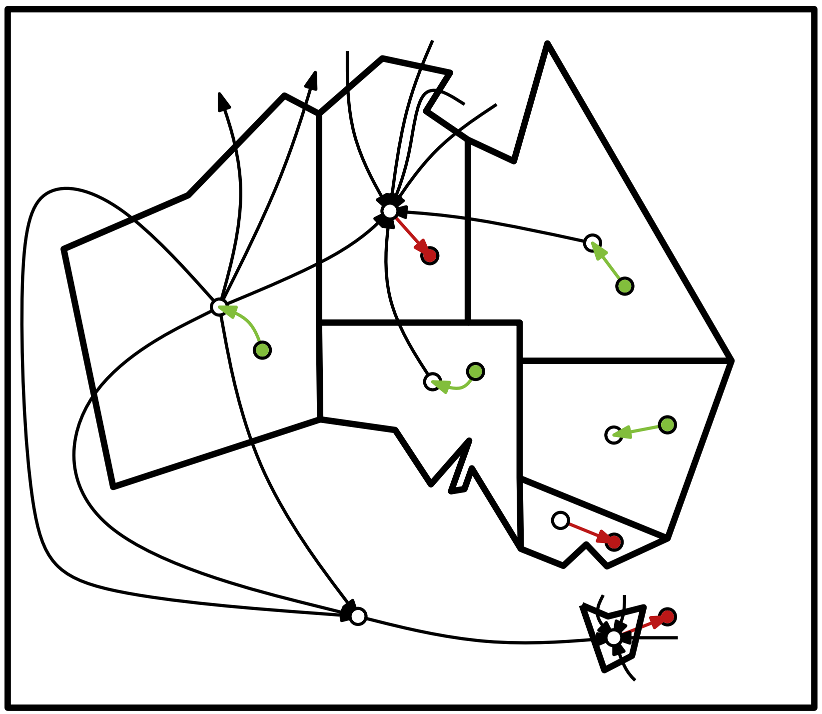


not all edges shown

- construct directed **dual graph**: one dual vertex v_i per face f_i, one dual edge per polygon edge
- **strong CAC:** direct edges outward if $|t_i - a_i| > 0$ & inward if $|t_i - a_i| < 0$
- **weak CAC:** bidirected pairs of edges
- add another vertex v_i' for each v_i
- if $\Delta_i = t_i - a_i > 0$ make v_i' a source and add (v_i', v_i) with capacity Δ_i
- if $\Delta_i = t_i - a_i < 0$ make v_i' a sink and add (v_i, v_i') with capacity $-\Delta_i$

Heuristic Algorithm

Idea: bending an edge transfers area from one face to another
$\rightarrow$ define a **network flow model** s.t. valid maximum flow corresponds to circular-arc cartogram with small error



not all edges shown

- construct directed **dual graph**:
  one dual vertex v_i per face f_i,
  one dual edge per polygon edge
- **strong CAC:** direct edges outward if $|t_i - a_i| > 0$ & inward if $|t_i - a_i| < 0$
- **weak CAC:** bidirected pairs of edges
- add another vertex v_i' for each v_i
- if $\Delta_i = t_i - a_i > 0$ make v_i' a source and add (v_i', v_i) with capacity Δ_i
- if $\Delta_i = t_i - a_i < 0$ make v_i' a sink and add (v_i, v_i') with capacity $-\Delta_i$

How do we set the remaining edge capacities?

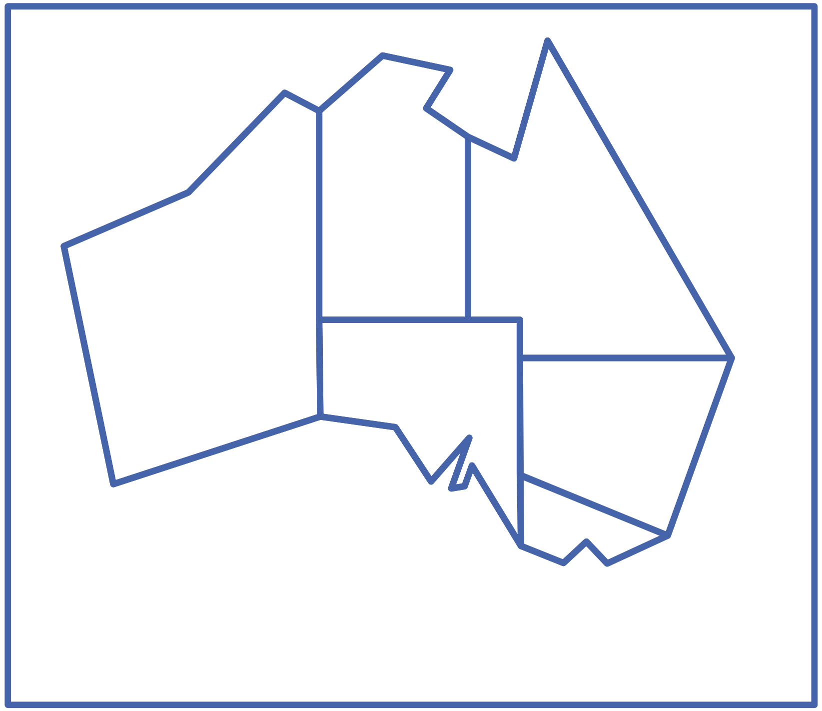
Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

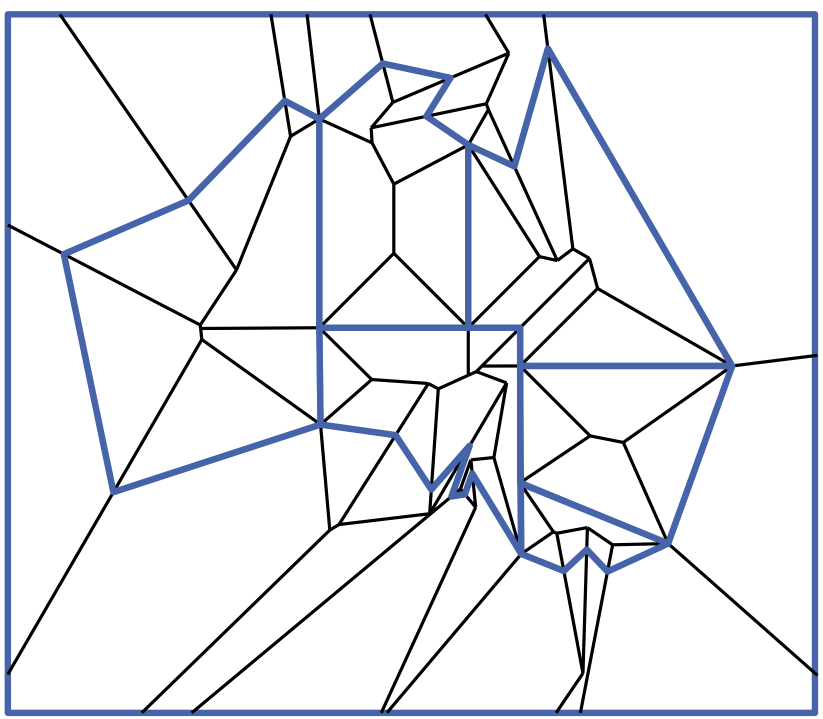
Solution: compute **straight skeleton** of the subdivision and confine every arc to stay within its skeleton cell



Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

Solution: compute **straight skeleton** of the subdivision and confine every arc to stay within its skeleton cell

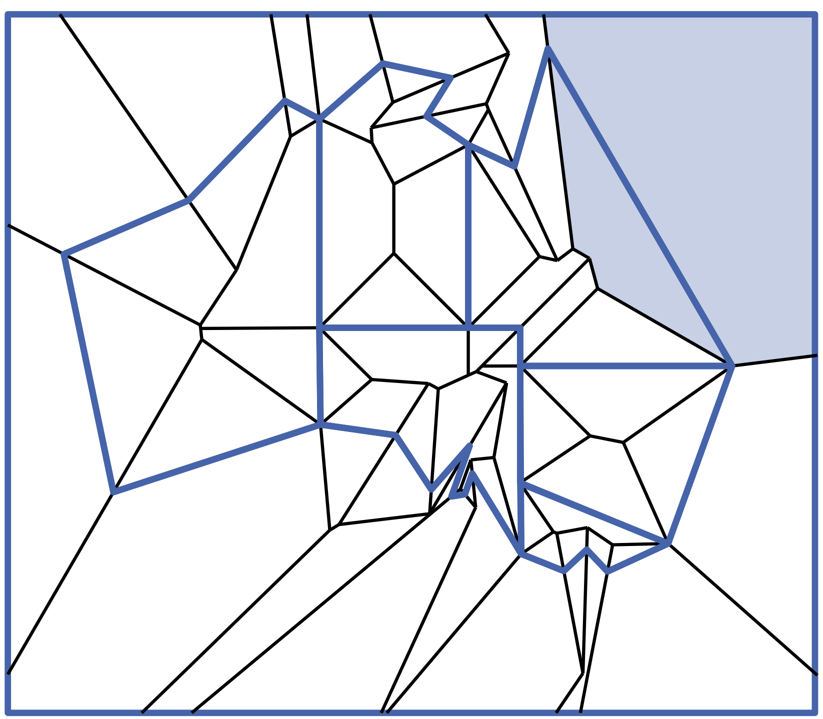


- straight skeleton can be computed in $O(m \log^2 m)$ time [Cheng, Vigneron '07]
- divides each m-edge polygon into m polygonal cells
- compute maximal arcs to both sides of each edge that remain inside their cells $\rightarrow$ no intersections
- max areas define edge capacities $c(e)$
- each flow $F(e) \leq c(e)$ can be realized as a valid circular arc with area transfer $F(e)$

Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

Solution: compute **straight skeleton** of the subdivision and confine every arc to stay within its skeleton cell

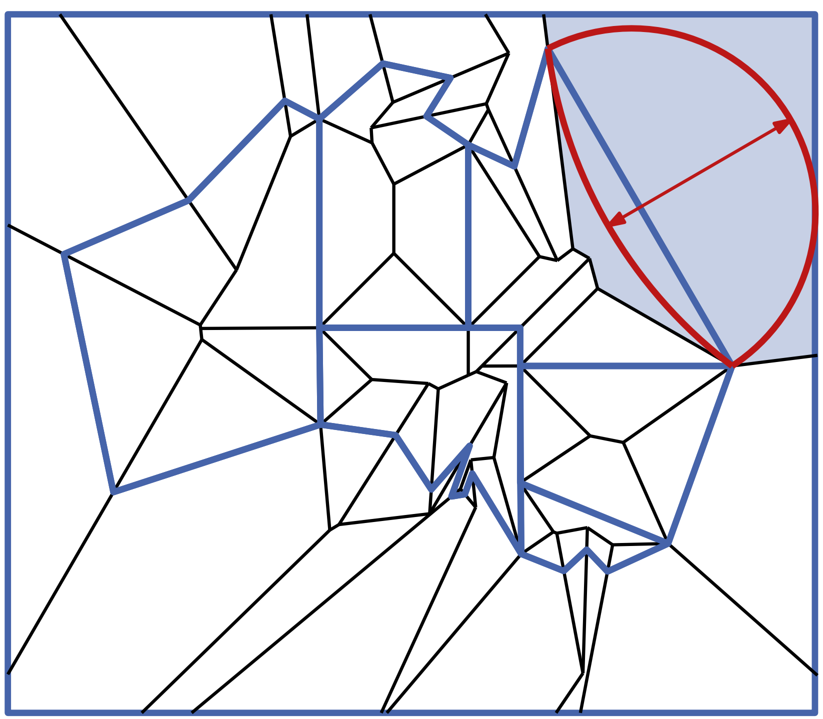


- straight skeleton can be computed in $O(m \log^2 m)$ time [Cheng, Vigneron '07]
- divides each m-edge polygon into m polygonal cells
- compute maximal arcs to both sides of each edge that remain inside their cells $\rightarrow$ no intersections
- max areas define edge capacities $c(e)$
- each flow $F(e) \leq c(e)$ can be realized as a valid circular arc with area transfer $F(e)$

Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

Solution: compute **straight skeleton** of the subdivision and confine every arc to stay within its skeleton cell

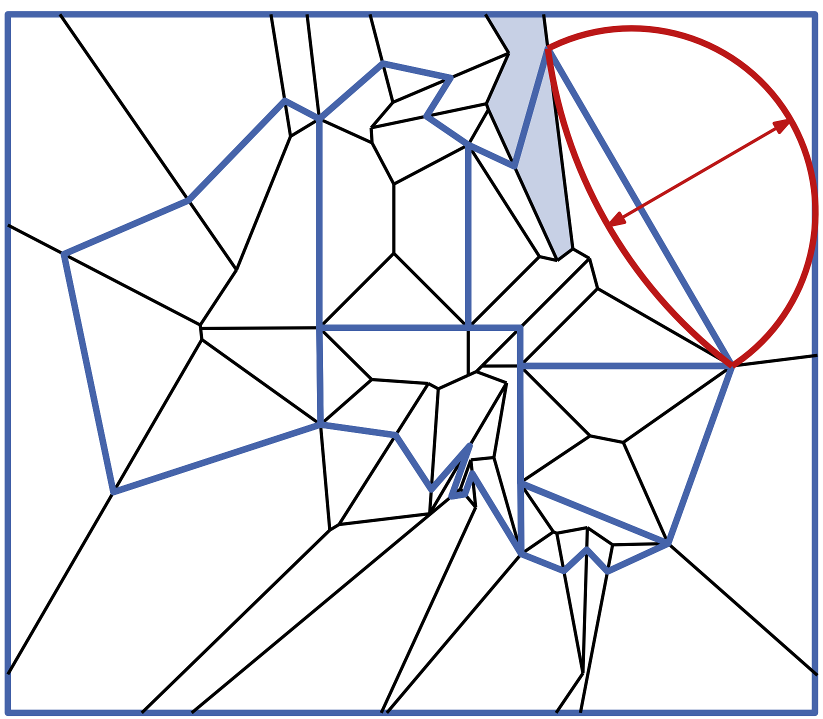


- straight skeleton can be computed in $O(m \log^2 m)$ time [Cheng, Vigneron '07]
- divides each m-edge polygon into m polygonal cells
- compute maximal arcs to both sides of each edge that remain inside their cells $\rightarrow$ no intersections
- max areas define edge capacities $c(e)$
- each flow $F(e) \leq c(e)$ can be realized as a valid circular arc with area transfer $F(e)$

Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

Solution: compute **straight skeleton** of the subdivision and confine every arc to stay within its skeleton cell

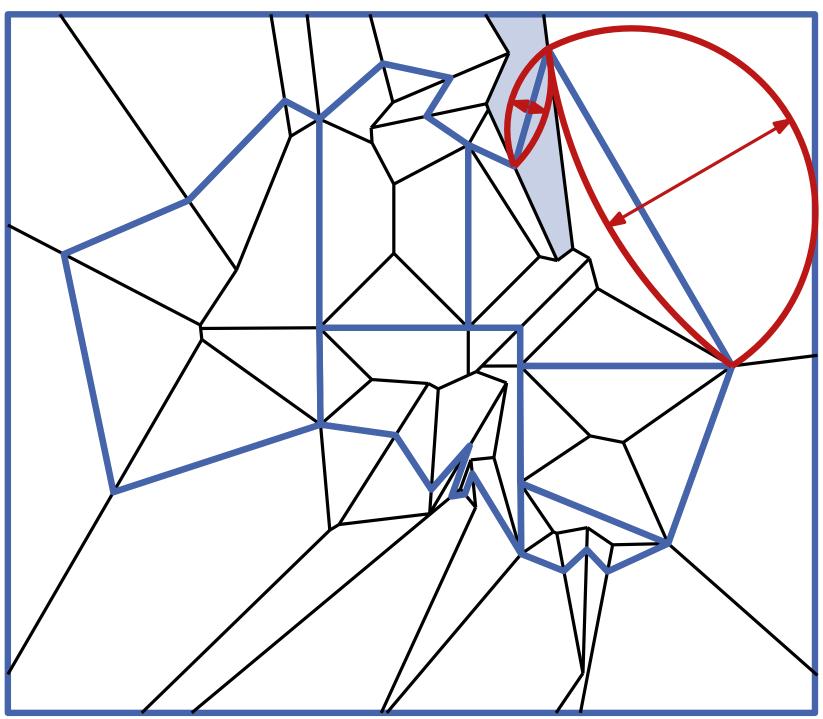


- straight skeleton can be computed in $O(m \log^2 m)$ time [Cheng, Vigneron '07]
- divides each m-edge polygon into m polygonal cells
- compute maximal arcs to both sides of each edge that remain inside their cells $\rightarrow$ no intersections
- max areas define edge capacities $c(e)$
- each flow $F(e) \leq c(e)$ can be realized as a valid circular arc with area transfer $F(e)$

Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

Solution: compute **straight skeleton** of the subdivision and confine every arc to stay within its skeleton cell

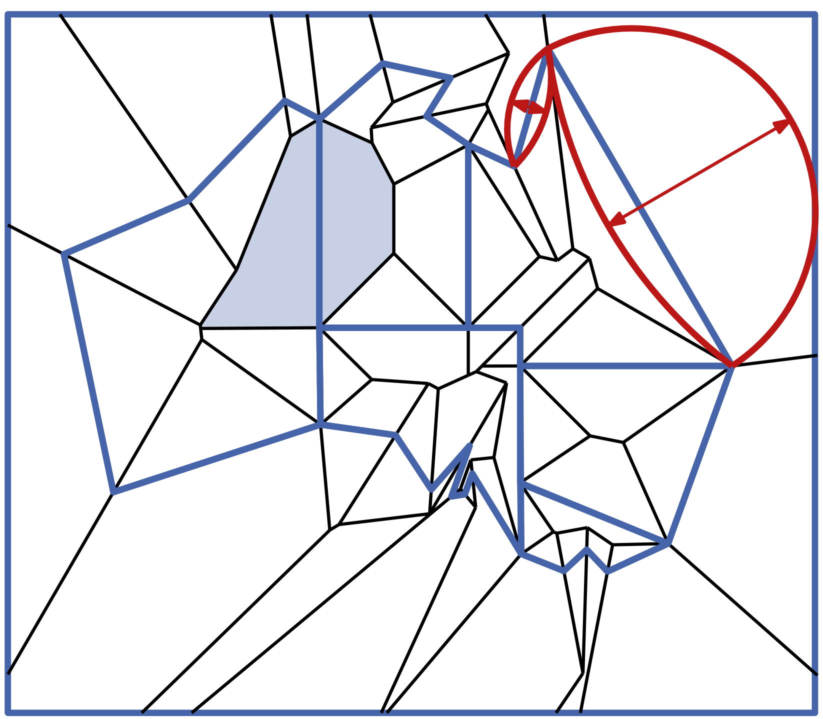


- straight skeleton can be computed in $O(m \log^2 m)$ time [Cheng, Vigneron '07]
- divides each m-edge polygon into m polygonal cells
- compute maximal arcs to both sides of each edge that remain inside their cells $\rightarrow$ no intersections
- max areas define edge capacities $c(e)$
- each flow $F(e) \leq c(e)$ can be realized as a valid circular arc with area transfer $F(e)$

Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

Solution: compute **straight skeleton** of the subdivision and confine every arc to stay within its skeleton cell

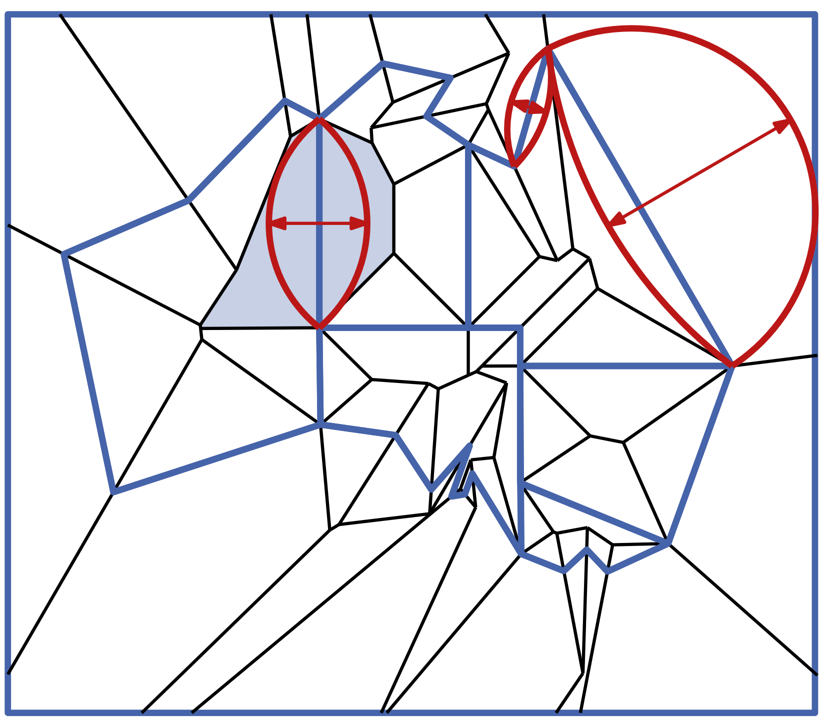


- straight skeleton can be computed in $O(m \log^2 m)$ time [Cheng, Vigneron '07]
- divides each m-edge polygon into m polygonal cells
- compute maximal arcs to both sides of each edge that remain inside their cells $\rightarrow$ no intersections
- max areas define edge capacities $c(e)$
- each flow $F(e) \leq c(e)$ can be realized as a valid circular arc with area transfer $F(e)$

Defining Edge Capacities

Requirement: regardless of flow value, corresponding circular arc is not allowed to intersect any other arc

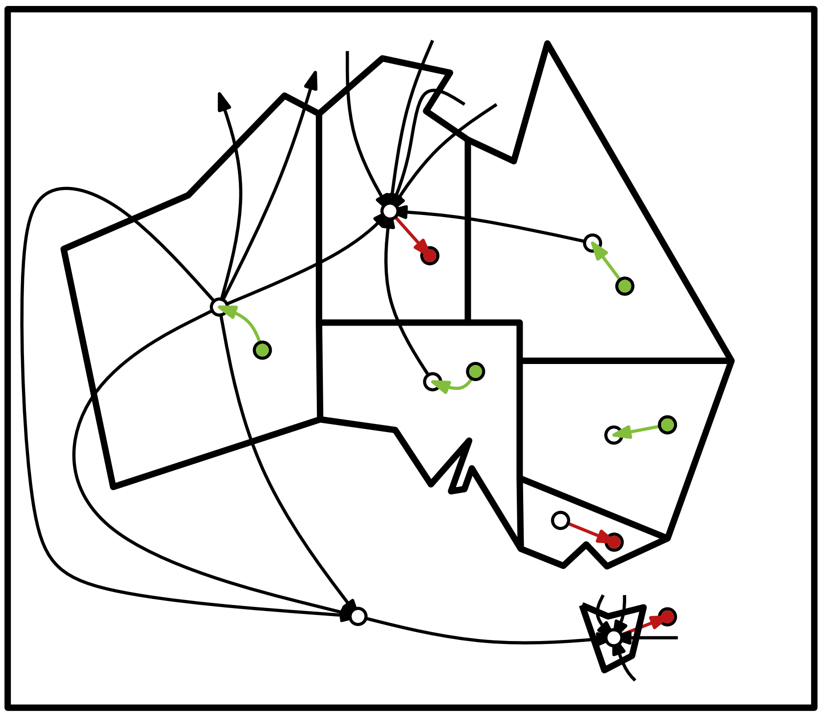
Solution: compute **straight skeleton** of the subdivision and confine every arc to stay within its skeleton cell



- straight skeleton can be computed in $O(m \log^2 m)$ time [Cheng, Vigneron '07]
- divides each m-edge polygon into m polygonal cells
- compute maximal arcs to both sides of each edge that remain inside their cells $\rightarrow$ no intersections
- max areas define edge capacities $c(e)$
- each flow $F(e) \leq c(e)$ can be realized as a valid circular arc with area transfer $F(e)$

Summary of the Algorithm

Via the dual graph G of the input subdivision and edge capacities c obtained from the straight skeleton we get a multiple-source multiple-sink flow network $\mathcal{N} = (G, c, S, T)$.

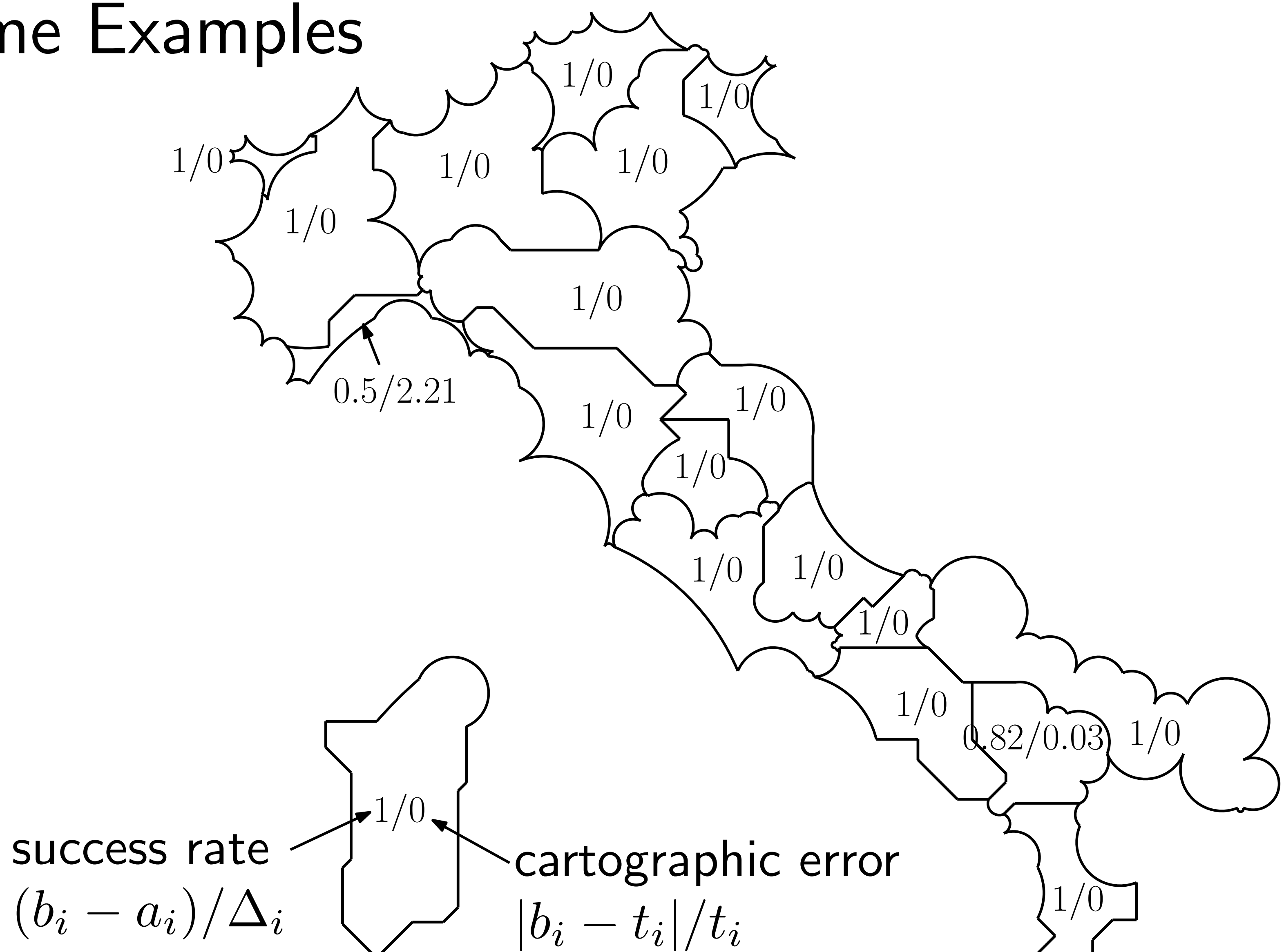


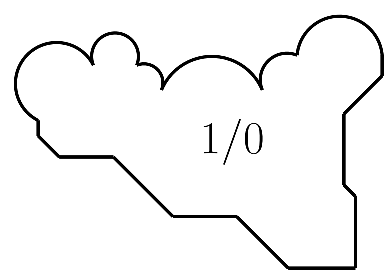
- G is planar as the dual graph of a planar graph
- maximum flow in a planar flow network $\mathcal{N}$ can be computed in $O(n \log^3 n)$ time [Borradaile et al. '11]
- since $\sum_i a_i = \sum_i t_i$ we get $\sum_{\Delta_i > 0} \Delta_i = -\sum_{\Delta_i < 0} \Delta_i$
- a maximum flow in $\mathcal{N}$ with value $D = \sum_{\Delta_i > 0} \Delta_i$ corresponds to an accurate circular-arc cartogram
- otherwise the cartographic error is minimized within the given capacity constraints

Some Examples

- prototype implementation in C++ using CGAL and Boost
- currently supports only weak cartograms
- measure for each face f_i
  - **success rate** $(b_i - a_i)/\Delta_i$
  - **cartographic error** $|b_i - t_i|/t_i$

Some Examples

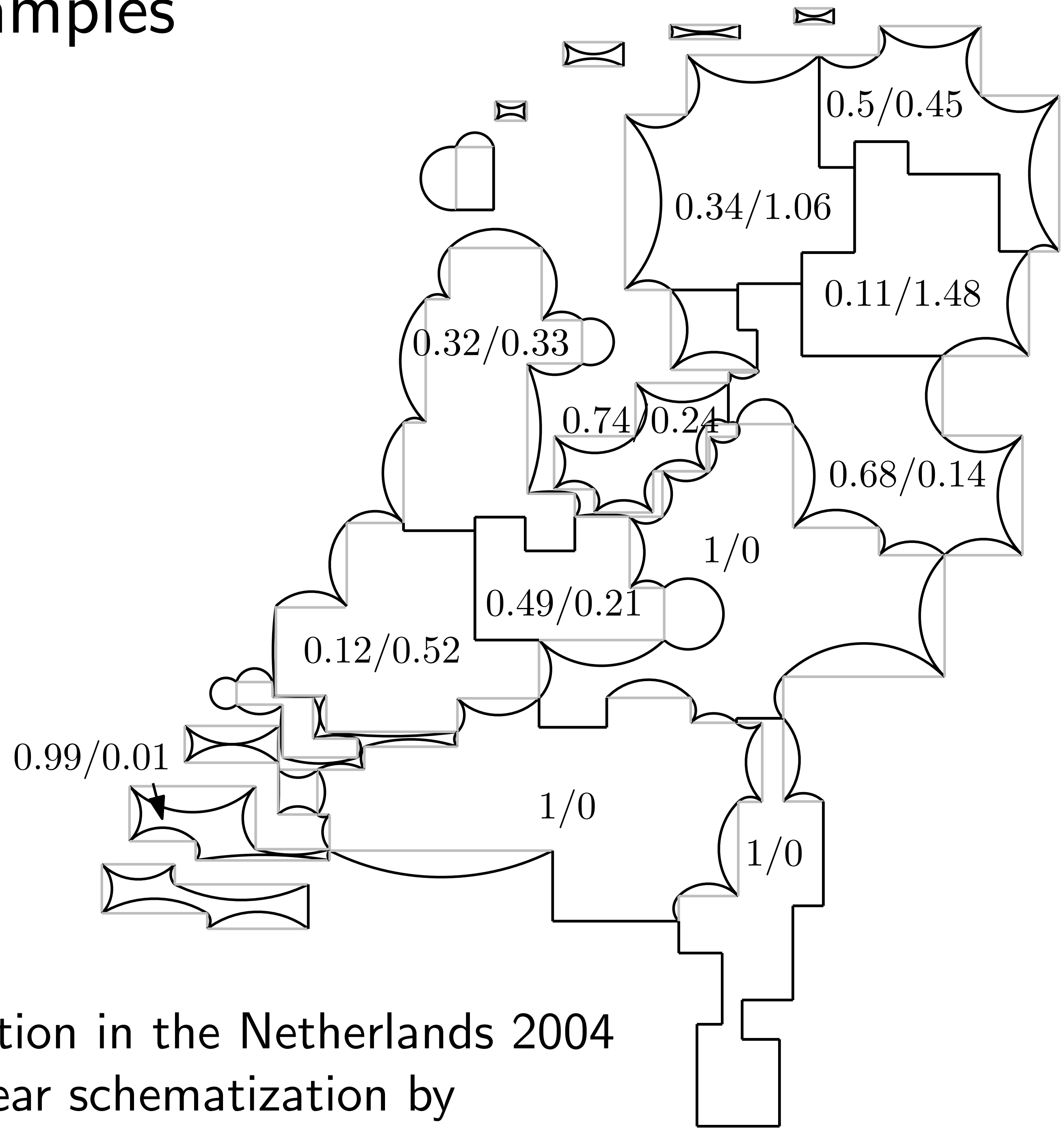




Data: agricultural use area in Italy 2010

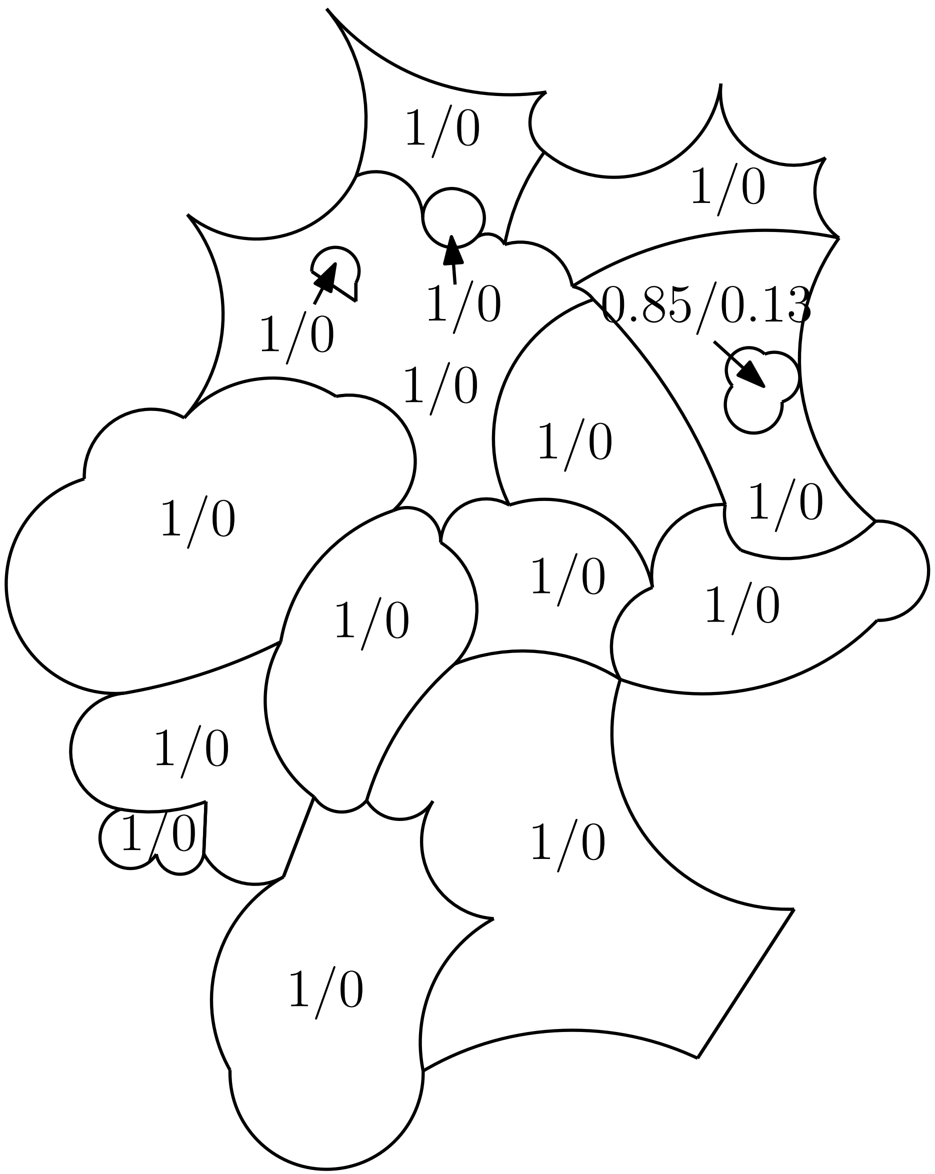
Map: octilinear schematization by [Buchin et al. '11]

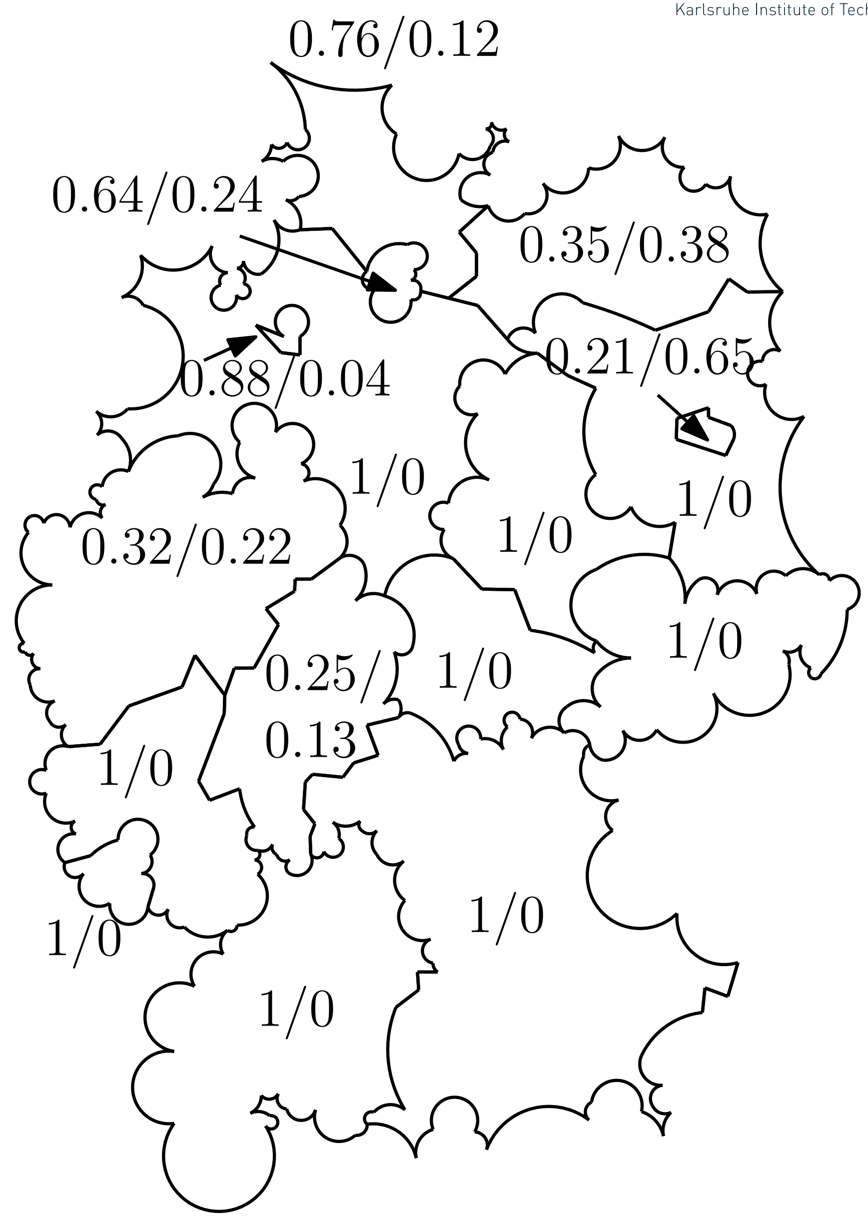
Some Examples



Data: population in the Netherlands 2004
Map: rectilinear schematization by
[Buchin et al. '11]

Some Examples





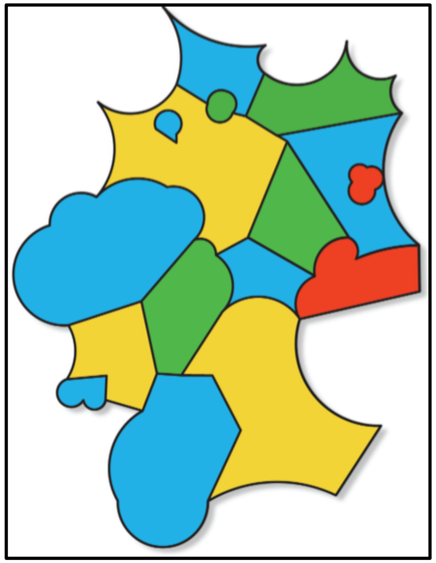
Data: railroad km in Germany 2010

Map: simplified manually (coarse and fine)

Zusammenfassung Kreisbogenkartogramme

Diskussion:

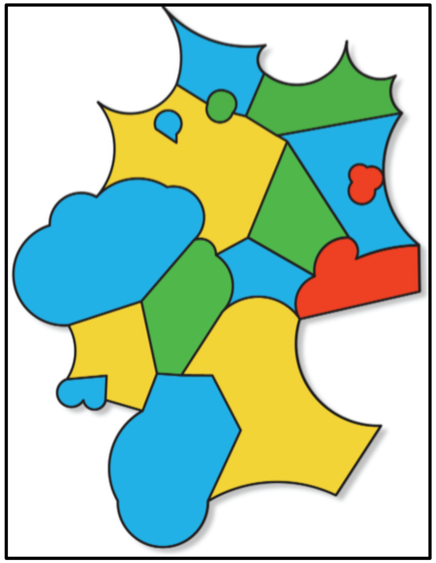
- Wiedererkennbarkeit der Form
- Flächenvergleichbarkeit
- Lage der Regionen
- korrekte Adjazenzen
- kleiner Flächenfehler
- geringe Komplexität
- Ablesen der Fläche



Zusammenfassung Kreisbogenkartogramme

Diskussion:

- Wiedererkennbarkeit der Form ○ ⊕
- Flächenvergleichbarkeit ⊖ ○
- Lage der Regionen ⊕ ○
- korrekte Adjazenzen ⊕
- kleiner Flächenfehler ○
- geringe Komplexität ○
- Ablesen der Fläche ⊖ ○



Zusammenfassung Kreisbogenkartogramme

Diskussion:

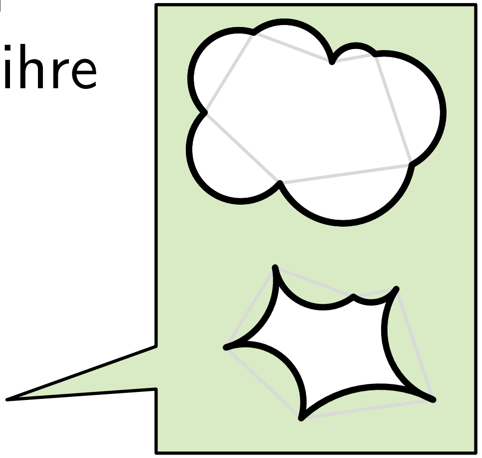
- Wiedererkennbarkeit der Form
- Flächenvergleichbarkeit
- Lage der Regionen
- korrekte Adjazenzen
- kleiner Flächenfehler
- geringe Komplexität
- Ablesen der Fläche

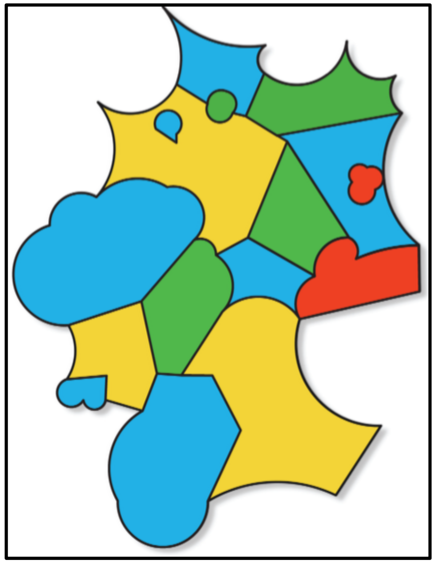
Weiteres Problem:

- umschlossene Regionen haben kaum oder gar kein Potential ihre Fläche zu ändern

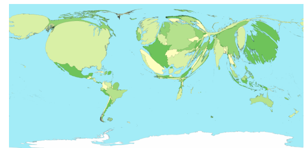
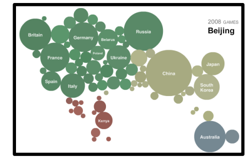
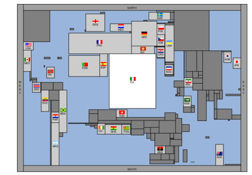
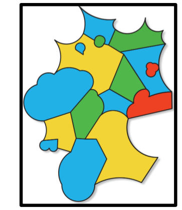
Weiterer Vorteil:

- Form der Region verdeutlicht Verhältnis Wert/Fläche

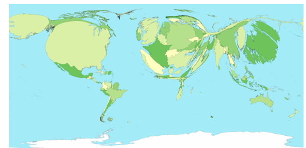
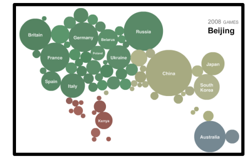
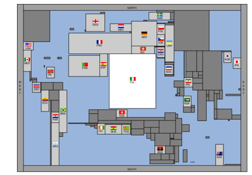
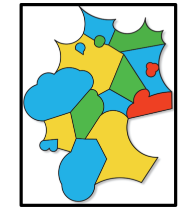




Zusammenfassung Kartogramme

				
Wiedererkennbarkeit	○	⊖	⊖	○ ⊕
Flächenvergleichbarkeit	⊖	⊕	⊕ ○	⊖ ○
Lage der Regionen	⊕ ○	⊖ ○	○	⊕ ○
korrekte Adjazenzen	⊕	⊖ ○	⊕/○	⊕
kleiner Flächenfehler	⊕ ○	⊕	○/⊕	○
geringe Komplexität	⊖	⊕	⊕	○
Ablesen der Fläche	⊖	⊕ ○	○	⊖ ○

Zusammenfassung Kartogramme

				
Wiedererkennbarkeit	○	⊖	⊖	○ ⊕
Flächenvergleichbarkeit	⊖	⊕	⊕ ○	⊖ ○
Lage der Regionen	⊕ ○	⊖ ○	○	⊕ ○
korrekte Adjazenzen	⊕	⊖ ○	⊕/○	⊕
kleiner Flächenfehler	⊕ ○	⊕	○/⊕	○
geringe Komplexität	⊖	⊕	⊕	○
Ablesen der Fläche	⊖	⊕ ○	○	⊖ ○

Kein Kartogrammtyp ist perfekt, alle haben ihre Vor- und Nachteile. Auswahl sollte je nach Anwendungszweck erfolgen.

Rektilineare Kartogramme

Ein Ausflug in die Theorie: Welche Polygonkomplexität braucht man, um jede Flächenzuweisung als rektilineares Kartogramm realisieren zu können?

Rektilineare Kartogramme

Ein Ausflug in die Theorie: Welche Polygonkomplexität braucht man, um jede Flächenzuweisung als rektilineares Kartogramm realisieren zu können?

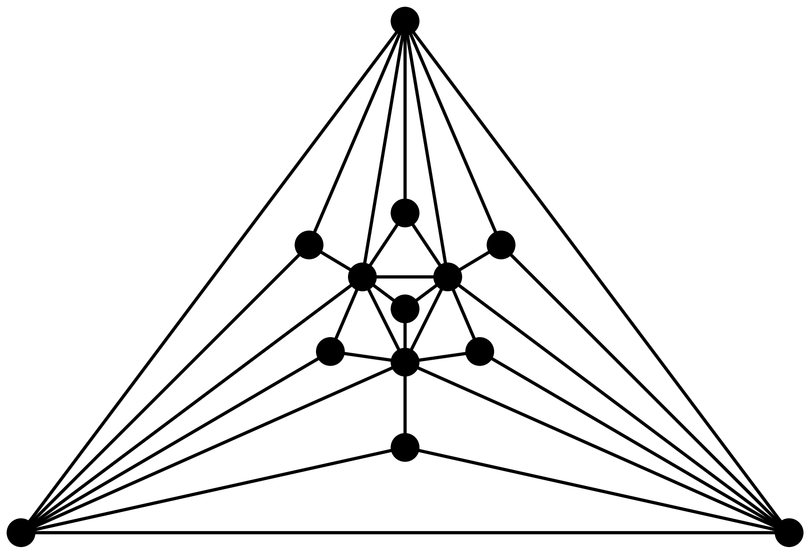
Satz: Es gibt planare triangulierte Graphen, die mindestens Komplexität 8 zur Kontaktrepräsentation mit rektilinearen Polygonen benötigen. [Yeap, Sarrafzadeh '93]

Rektilineare Kartogramme

Ein Ausflug in die Theorie: Welche Polygonkomplexität braucht man, um jede Flächenzuweisung als rektilineares Kartogramm realisieren zu können?

Satz: Es gibt planare triangulierte Graphen, die mindestens Komplexität 8 zur Kontaktrepräsentation mit rektilinearen Polygonen benötigen. [Yeap, Sarrafzadeh '93]

Beweis:

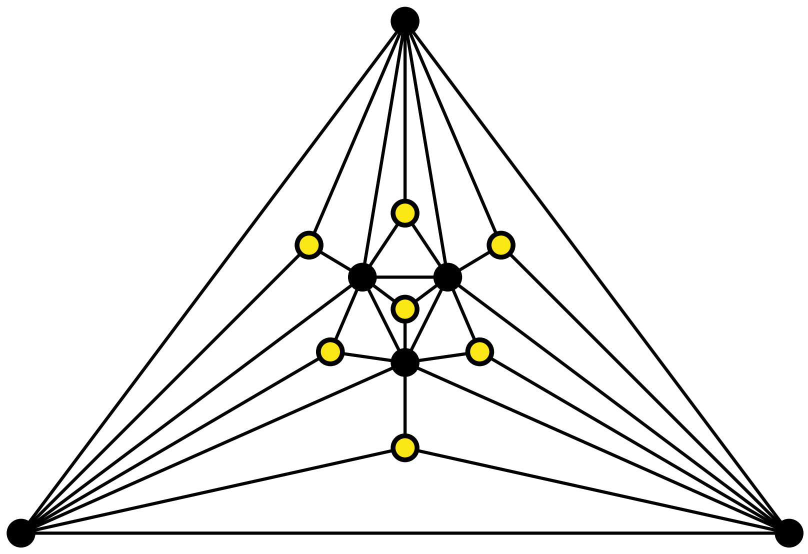


Rektilineare Kartogramme

Ein Ausflug in die Theorie: Welche Polygonkomplexität braucht man, um jede Flächenzuweisung als rektilineares Kartogramm realisieren zu können?

Satz: Es gibt planare triangulierte Graphen, die mindestens Komplexität 8 zur Kontaktrepräsentation mit rektilinearen Polygonen benötigen. [Yeap, Sarrafzadeh '93]

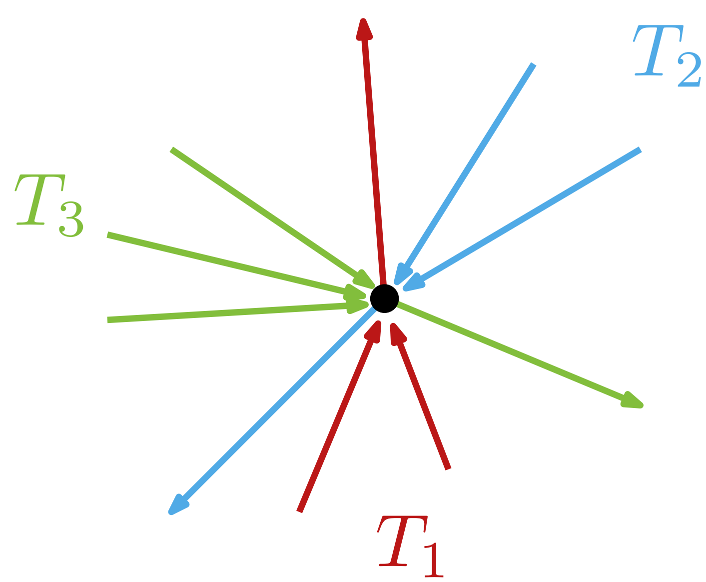
Beweis:



Schnyder Realizer

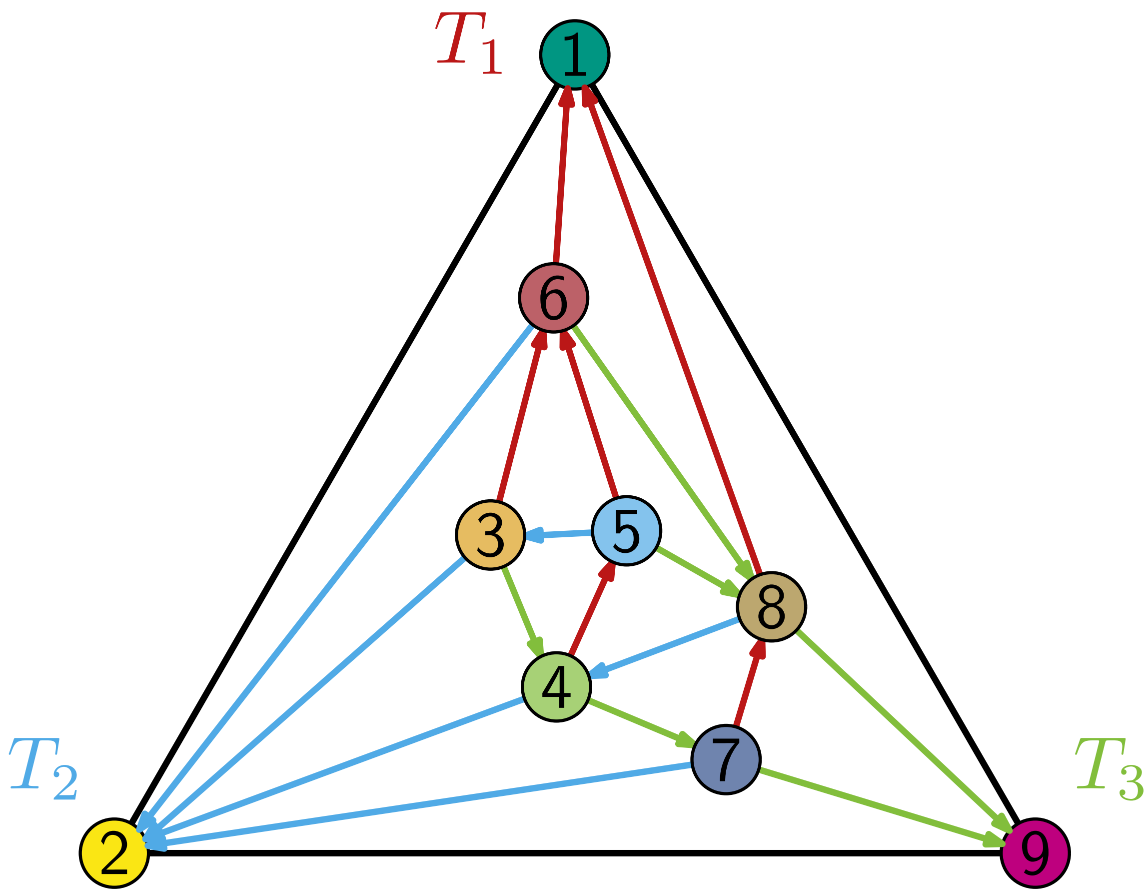
Sei G ein triangulierter planarer Graph. Ein **Schnyder Realizer** partitioniert die internen Kanten in drei Mengen T_1, T_2, T_3 von gerichteten Kanten, so dass

- jeder innere Knoten v hat genau eine Kante in jedem T_i^{out}
- die Ordnung der Kanten um jeden Knoten v im GUZS ist $T_1^{\text{in}}, T_3^{\text{out}}, T_2^{\text{in}}, T_1^{\text{out}}, T_3^{\text{in}}, T_2^{\text{out}}$



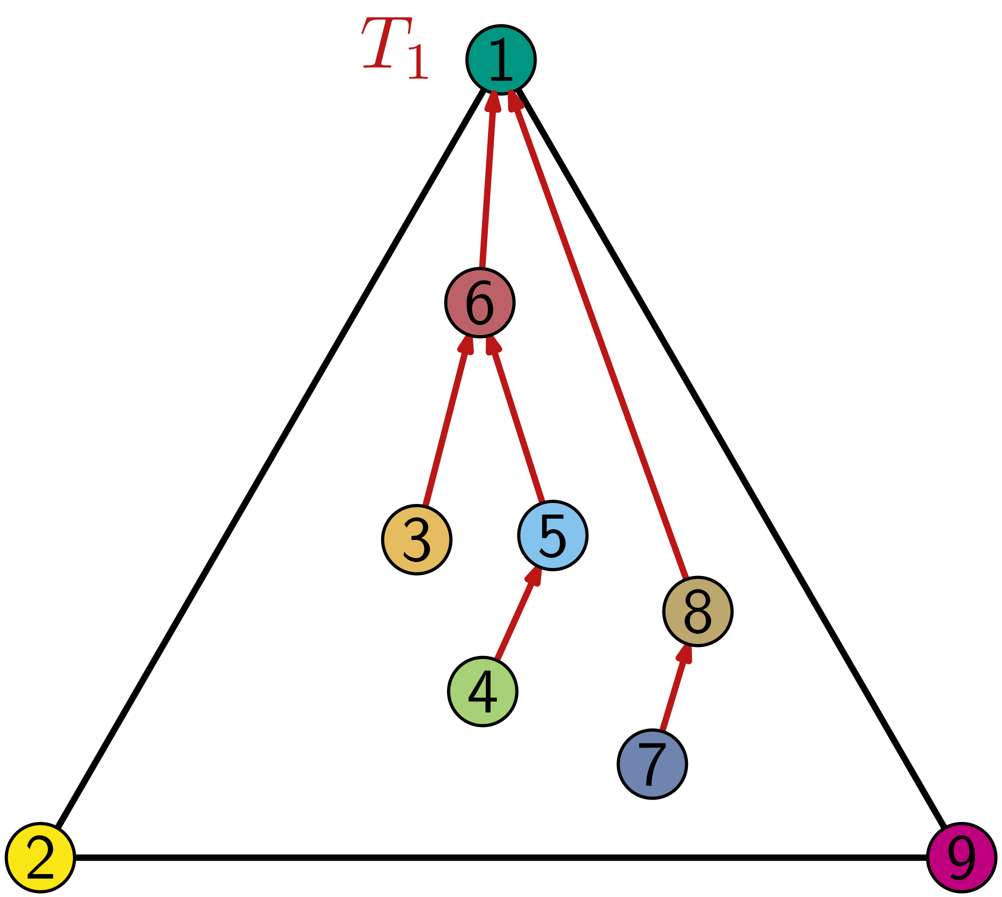
Schnyder Realizer

Satz: Jede Menge T_i $(i = 1, 2, 3)$ ist ein Spannbaum aller inneren Knoten und eines äußeren Knotens.
Jeder triangulierte Graph besitzt einen Schnyder Realizer und dieser kann in $O(n)$ Zeit berechnet werden.



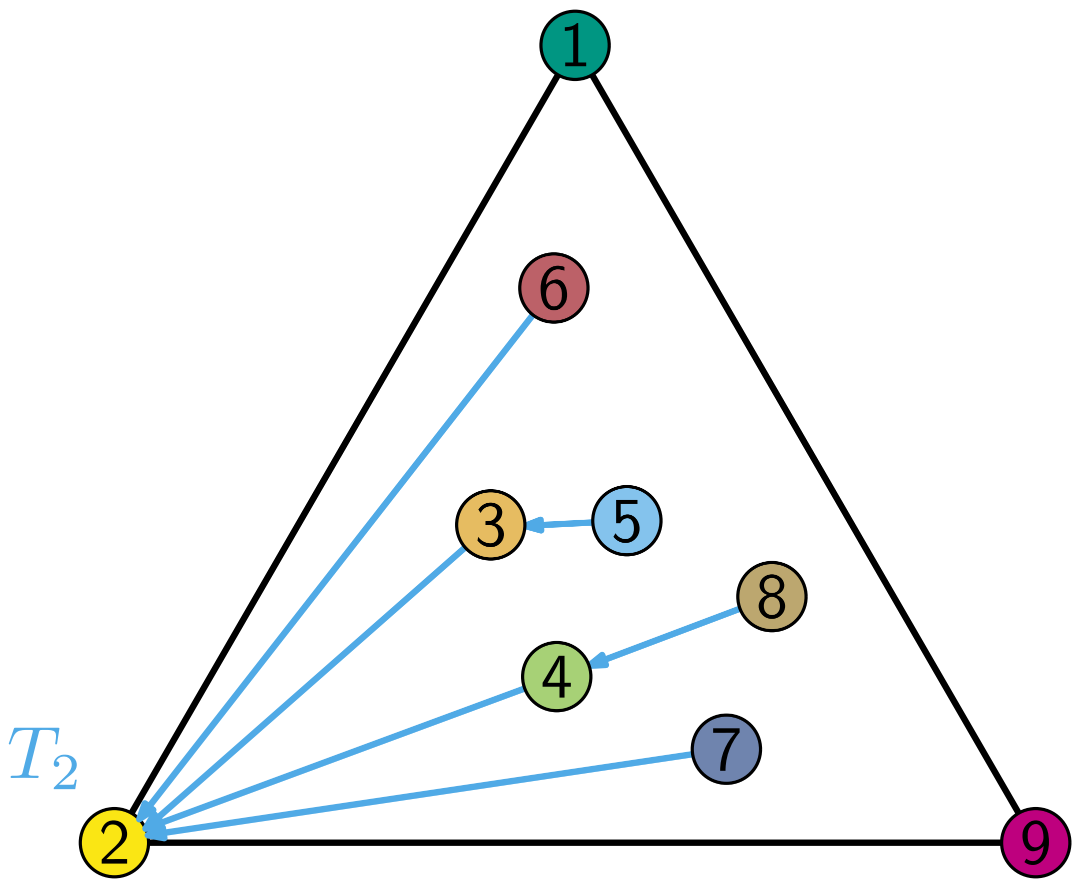
Schnyder Realizer

Satz: Jede Menge T_i $(i = 1, 2, 3)$ ist ein Spannbaum aller inneren Knoten und eines äußeren Knotens.
Jeder triangulierte Graph besitzt einen Schnyder Realizer und dieser kann in $O(n)$ Zeit berechnet werden.



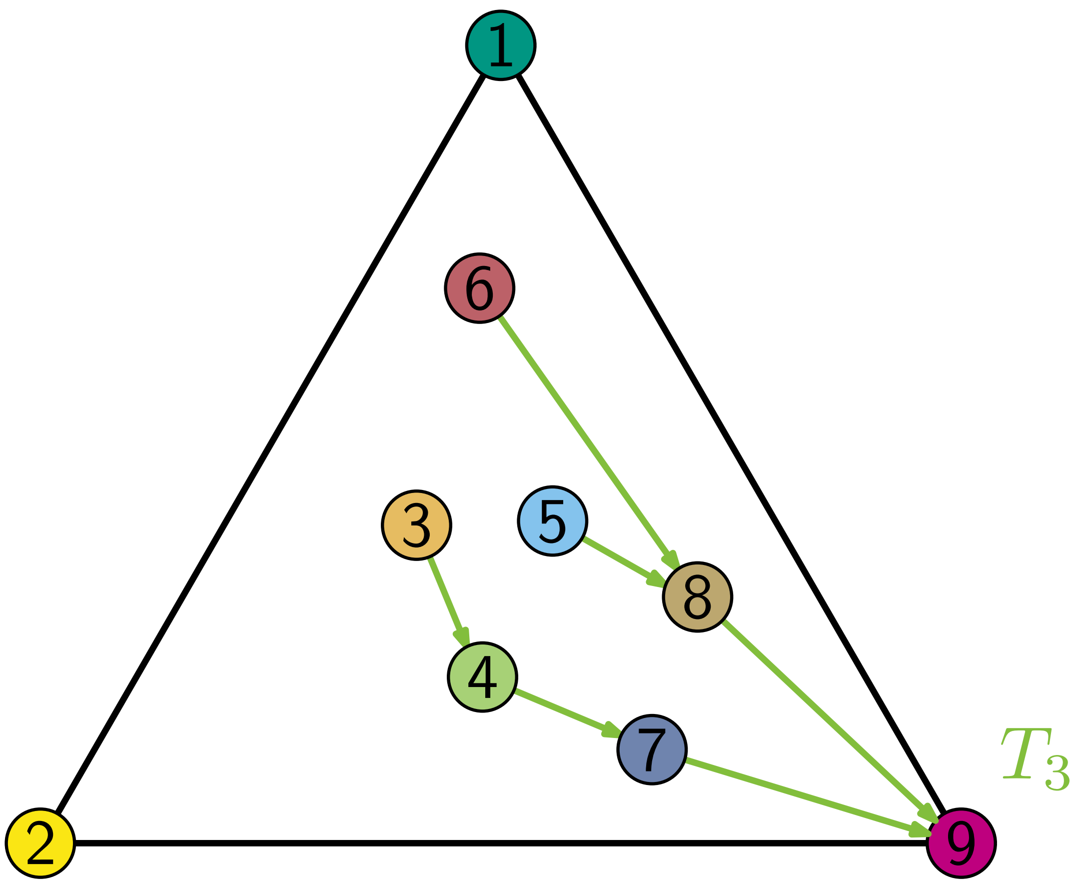
Schnyder Realizer

Satz: Jede Menge T_i $(i = 1, 2, 3)$ ist ein Spannbaum aller inneren Knoten und eines äußeren Knotens.
Jeder triangulierte Graph besitzt einen Schnyder Realizer und dieser kann in $O(n)$ Zeit berechnet werden.



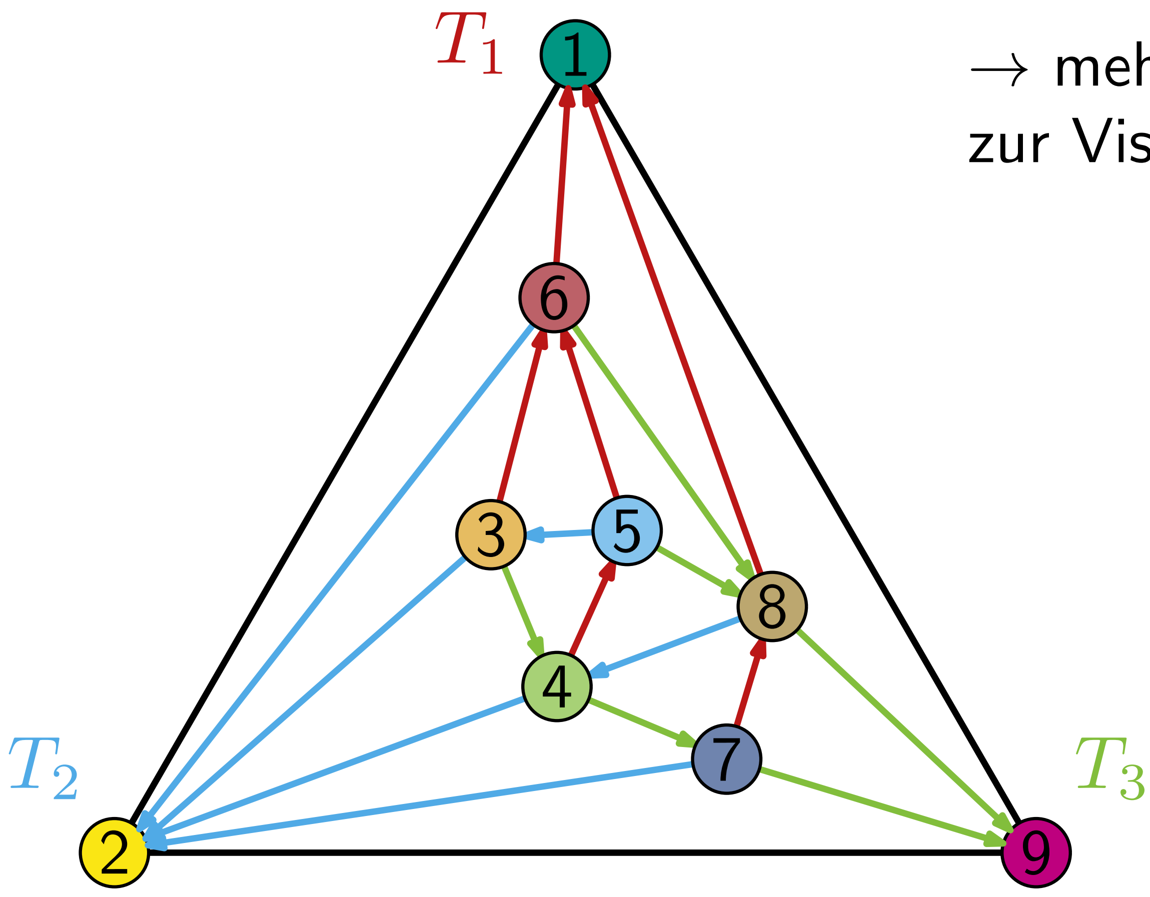
Schnyder Realizer

Satz: Jede Menge T_i $(i = 1, 2, 3)$ ist ein Spannbaum aller inneren Knoten und eines äußeren Knotens.
Jeder triangulierte Graph besitzt einen Schnyder Realizer und dieser kann in $O(n)$ Zeit berechnet werden.



Schnyder Realizer

Satz: Jede Menge T_i $(i = 1, 2, 3)$ ist ein Spannbaum aller inneren Knoten und eines äußeren Knotens.
Jeder triangulierte Graph besitzt einen Schnyder Realizer und dieser kann in $O(n)$ Zeit berechnet werden.



$\rightarrow$ mehr dazu in der VL Algorithmen zur Visualisierung von Graphen

Kanonische Ordnung

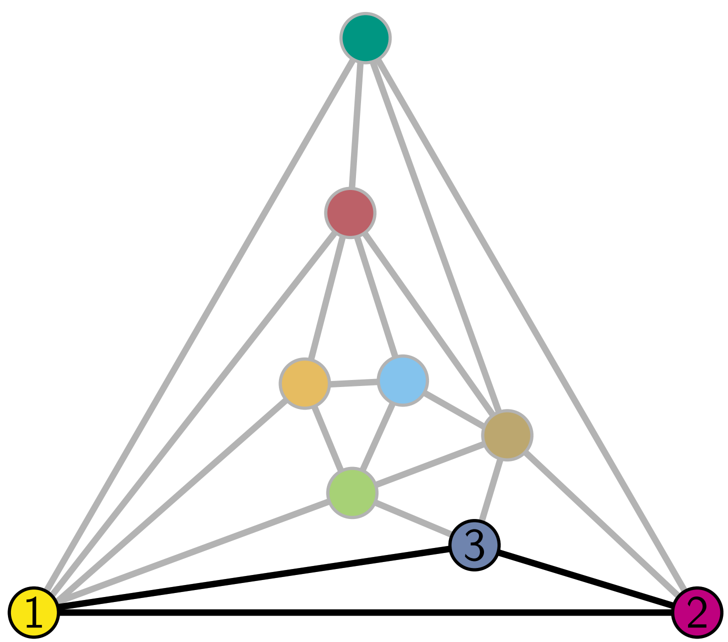
Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$

Kanonische Ordnung

Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

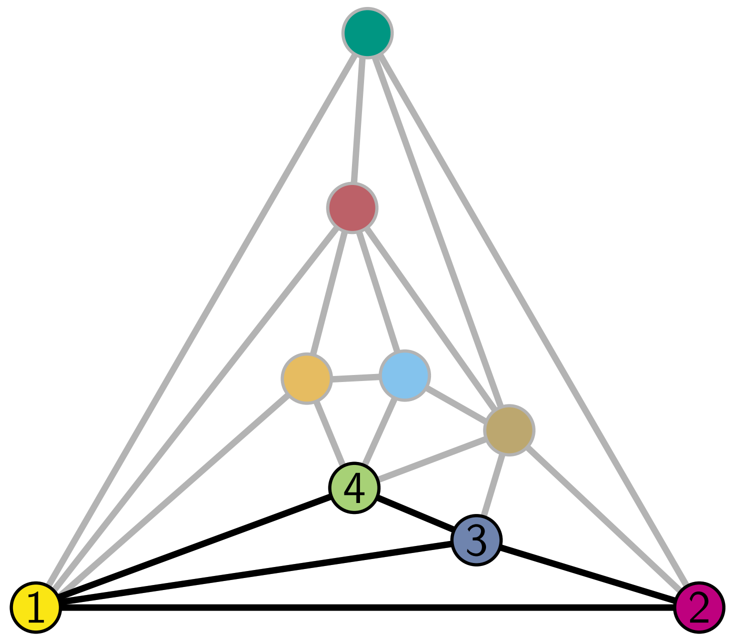
- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$



Kanonische Ordnung

Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

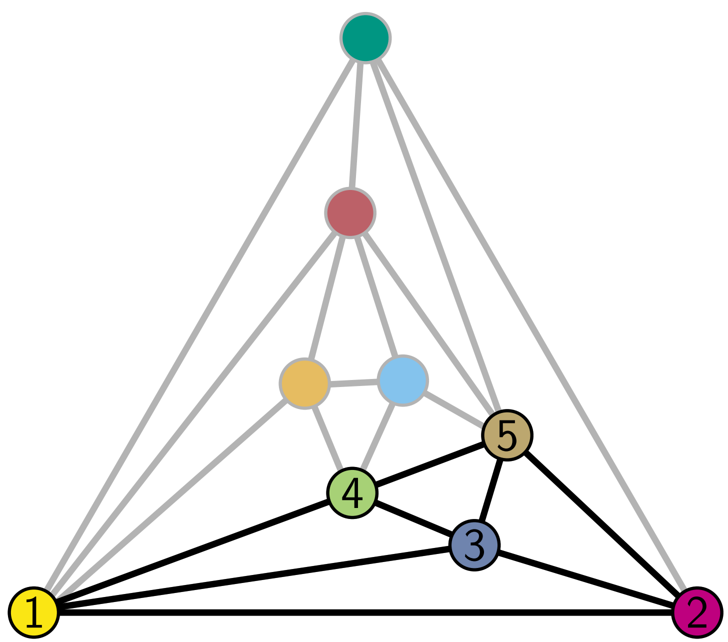
- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$



Kanonische Ordnung

Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \dots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

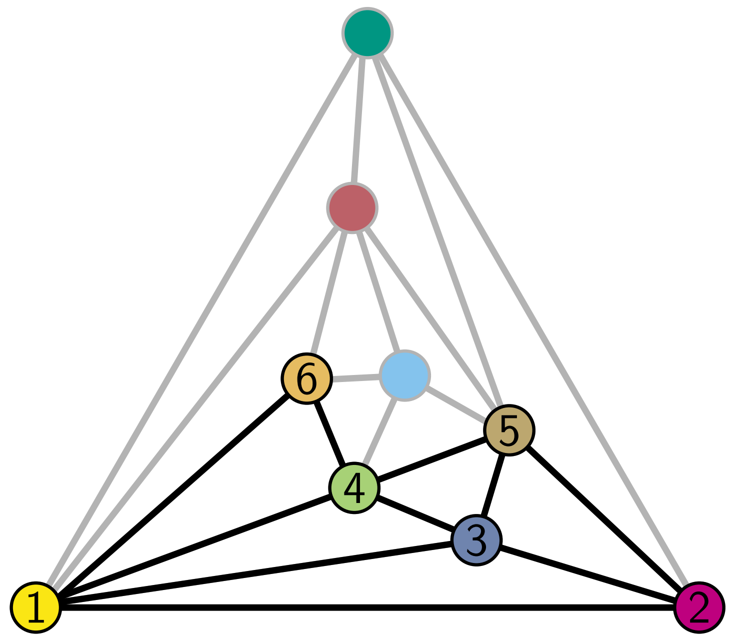
- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \dots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$



Kanonische Ordnung

Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

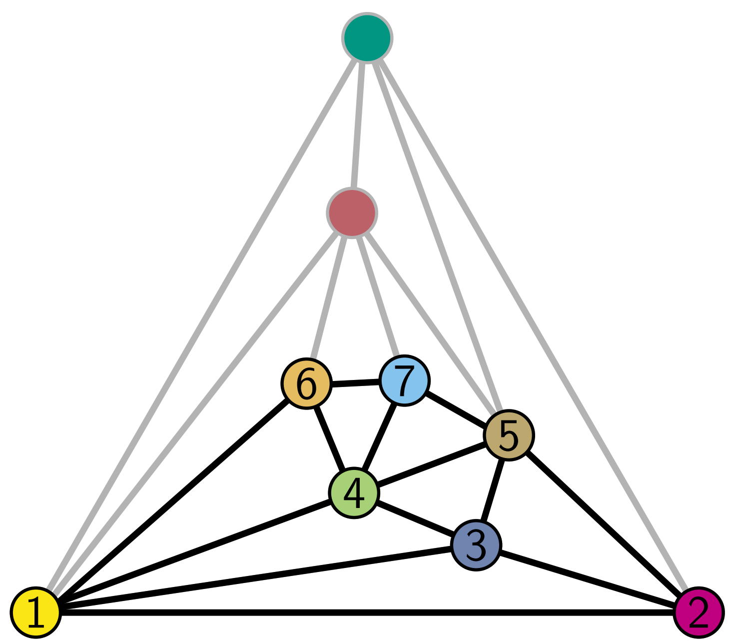
- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$



Kanonische Ordnung

Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

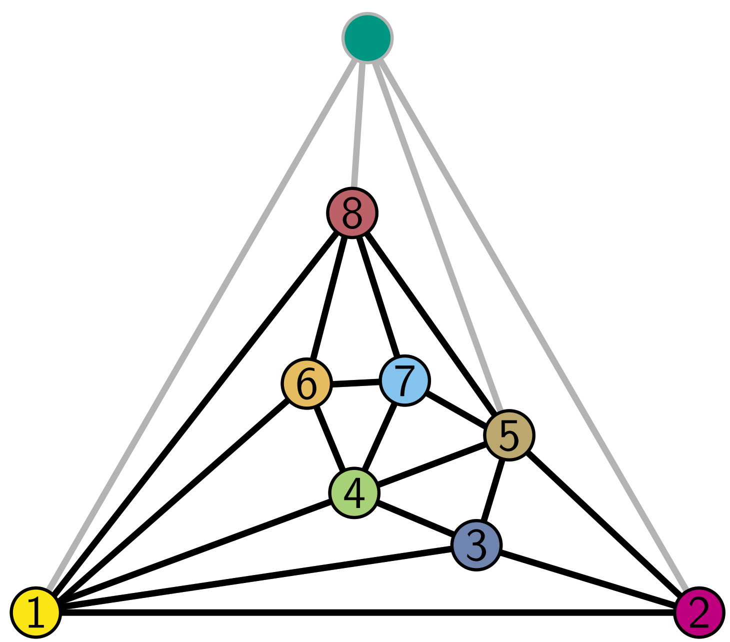
- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$



Kanonische Ordnung

Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

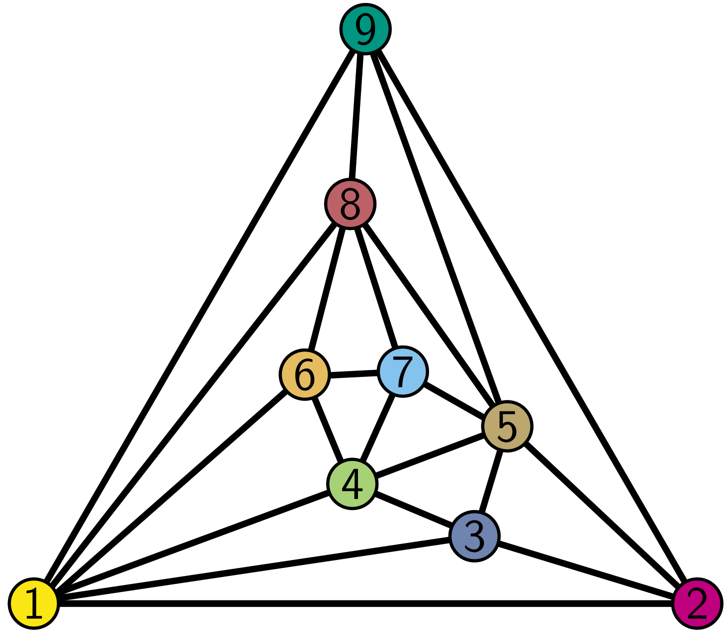
- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$



Kanonische Ordnung

Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

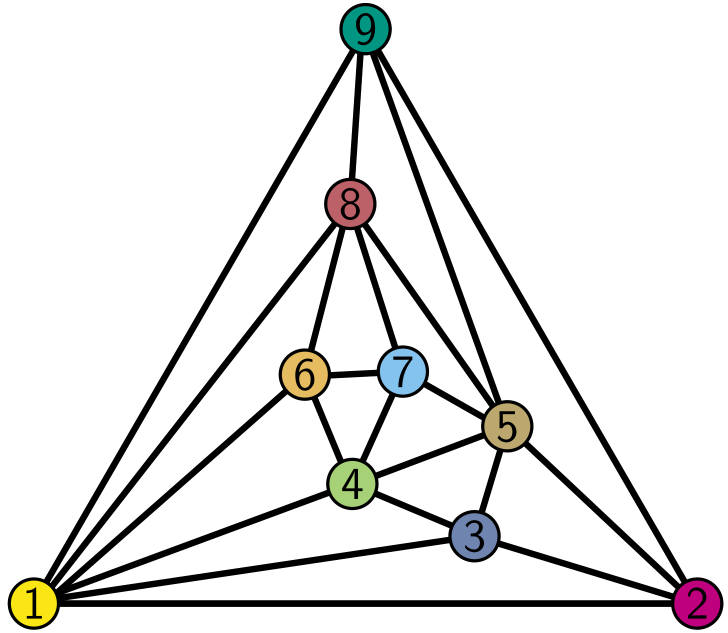
- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$



Kanonische Ordnung

Die **kanonische Ordnung** der Knoten eines triangulierten planaren Graphen G mit den drei äußeren Knoten u, v, w im GUZS ist eine Ordnung $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$, so dass für jedes i $(4 \leq i \leq n)$ gilt:

- Graph $G_{i-1} \subseteq G$ induziert durch $v_1, \ldots, v_{i-1}$ ist 2-fach zshgd. und äußere Facette ist begrenzt durch Kreis C_{i-1}, der $v_1 v_2$ enthält
- Knoten v_i liegt in der äußeren Facette von G_{i-1} und seine Nachbarn in G_{i-1} bilden zusammenhängendes Intervall im Pfad $C_{i-1} \setminus v_1 v_2$

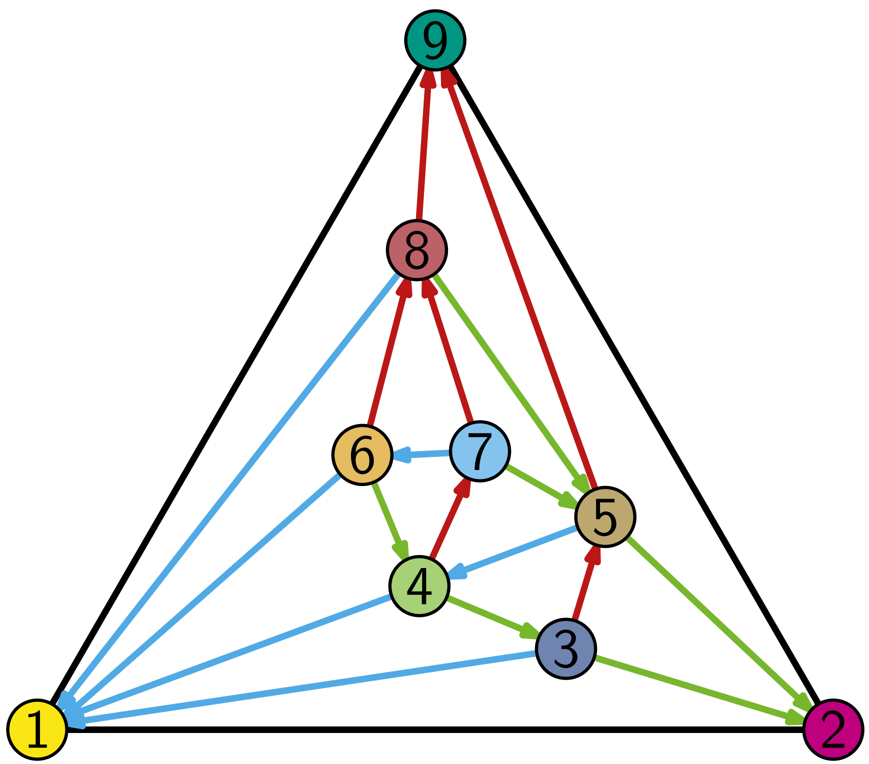


Satz: Jeder triangulierte Graph G besitzt eine kanonische Ordnung; sie kann in $O(n)$ Zeit berechnet werden.

$\rightarrow$ mehr dazu in der VL Algorithmen zur Visualisierung von Graphen

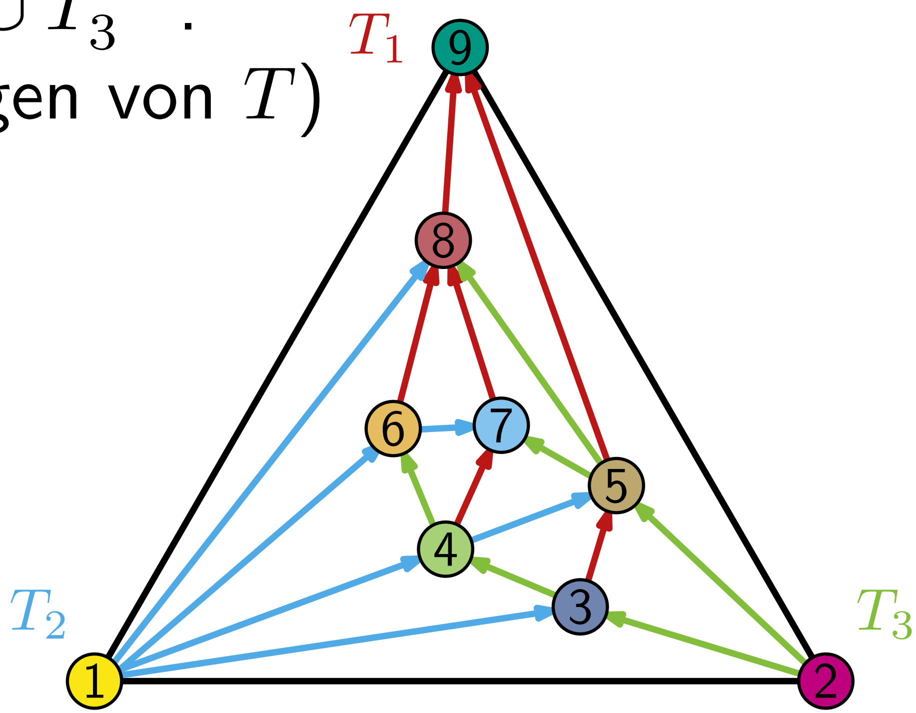
Kanonische Ordnung & Schnyder Realizer

- Eine kanonische Ordnung der Knoten von G definiert einen Schnyder Realizer von G, indem jedem Knoten v_i drei ausgehende Kanten zum ersten und letzten Knoten in C_{i-1} und zum Nachfolger mit höchstem Index zugeordnet werden.



Kanonische Ordnung & Schnyder Realizer

- Eine kanonische Ordnung der Knoten von G definiert einen Schnyder Realizer von G, indem jedem Knoten v_i drei ausgehende Kanten zum ersten und letzten Knoten in C_{i-1} und zum Nachfolger mit höchstem Index zugeordnet werden.
- Ein Schnyder Realizer mit Bäumen T_1, T_2, T_3 definiert eine kanonische Ordnung als topologische Ordnung des azyklischen Graphen $T_1 \cup T_2^{-1} \cup T_3^{-1}$.
  (T^{-1}: invertiere Kantenrichtungen von T)



8-seitige rektilineare Kartogramme [Alam et al. '11]

Algorithmus hat drei Phasen:

- erzeuge T-Kontaktrepräsentation
- wandle jedes T in T-förmiges Polygon um
- weise verbleibende Löcher den T-Polygonen zu

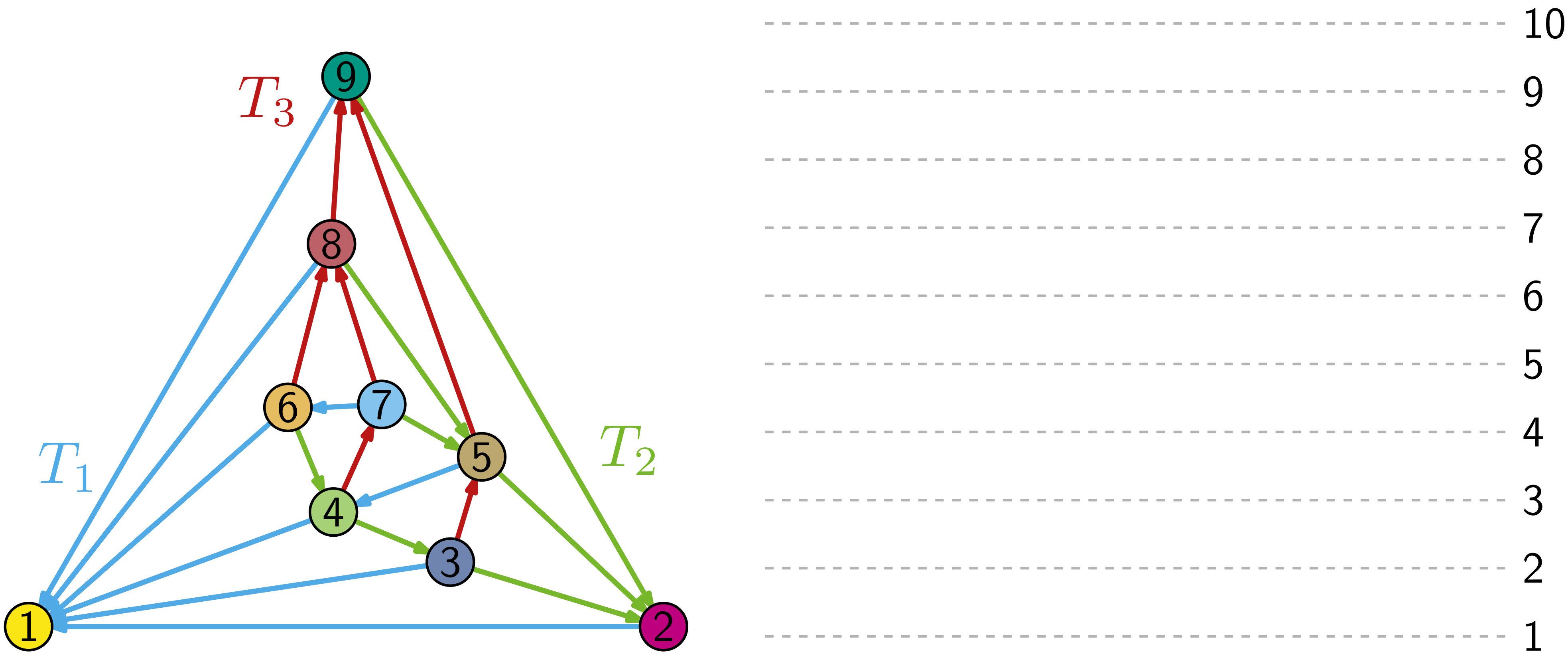
8-seitige rektilineare Kartogramme [Alam et al. '11]

Algorithmus hat drei Phasen:

- erzeuge T-Kontaktrepräsentation
- wandle jedes T in T-förmiges Polygon um
- weise verbleibende Löcher den T-Polygonen zu

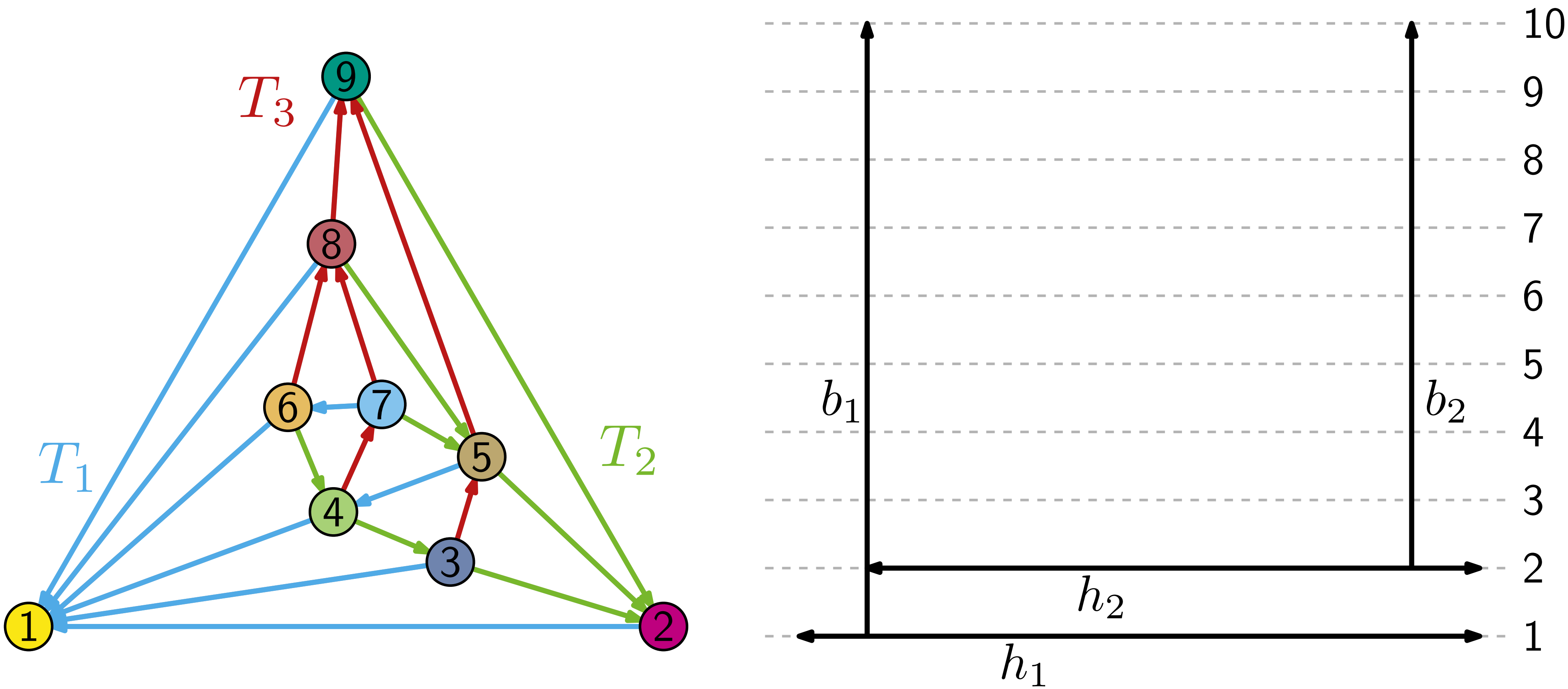
Wir zeigen:
Das entstandene rektilineare Layout ist flächenuniversell und somit existiert immer ein korrektes Kartogramm.

Phase 1: T-Kontaktrepräsentation



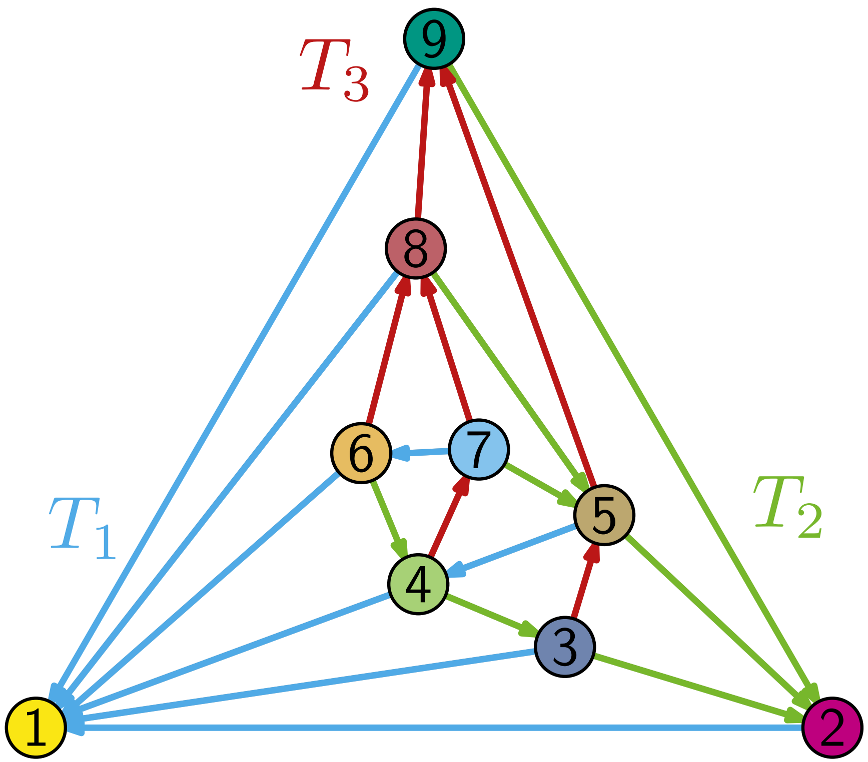
- betrachte Knoten in kanonischer Ordnung und zug. Schnyder Realizer
- $\Phi_i(k)$ sei Vater in T_i des Knotens v_k

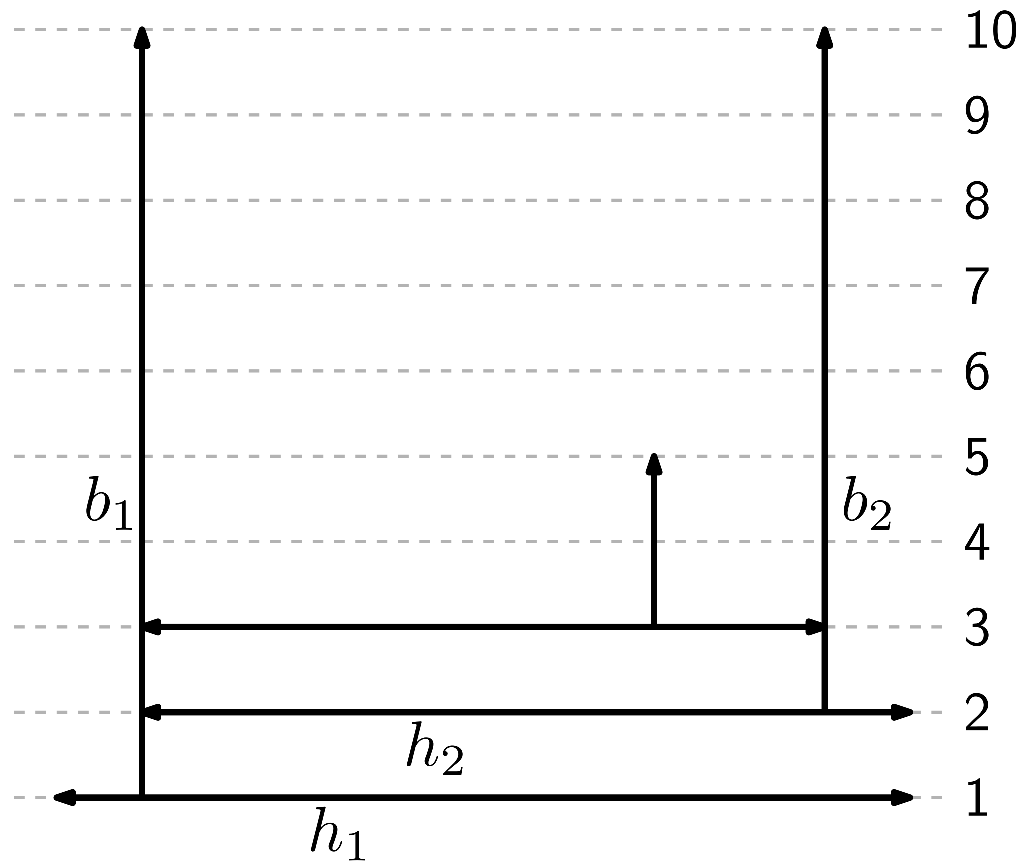
Phase 1: T-Kontaktrepräsentation



- betrachte Knoten in kanonischer Ordnung und zug. Schnyder Realizer
- $\Phi_i(k)$ sei Vater in T_i des Knotens v_k
- setze T's für v_1 und v_2 auf $y = 1$ und $y = 2$

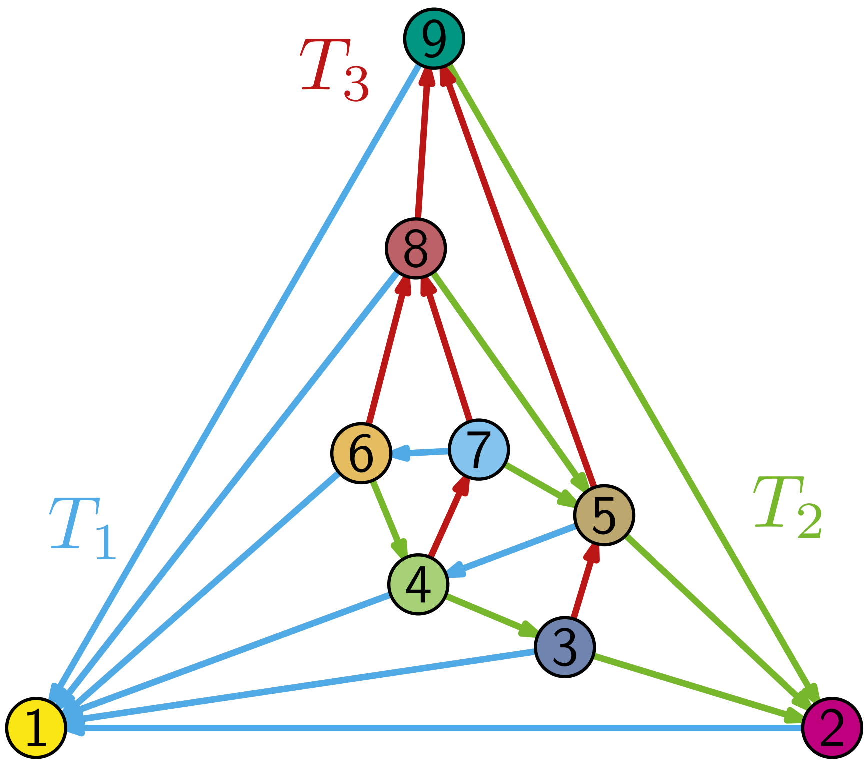
Phase 1: T-Kontaktrepräsentation

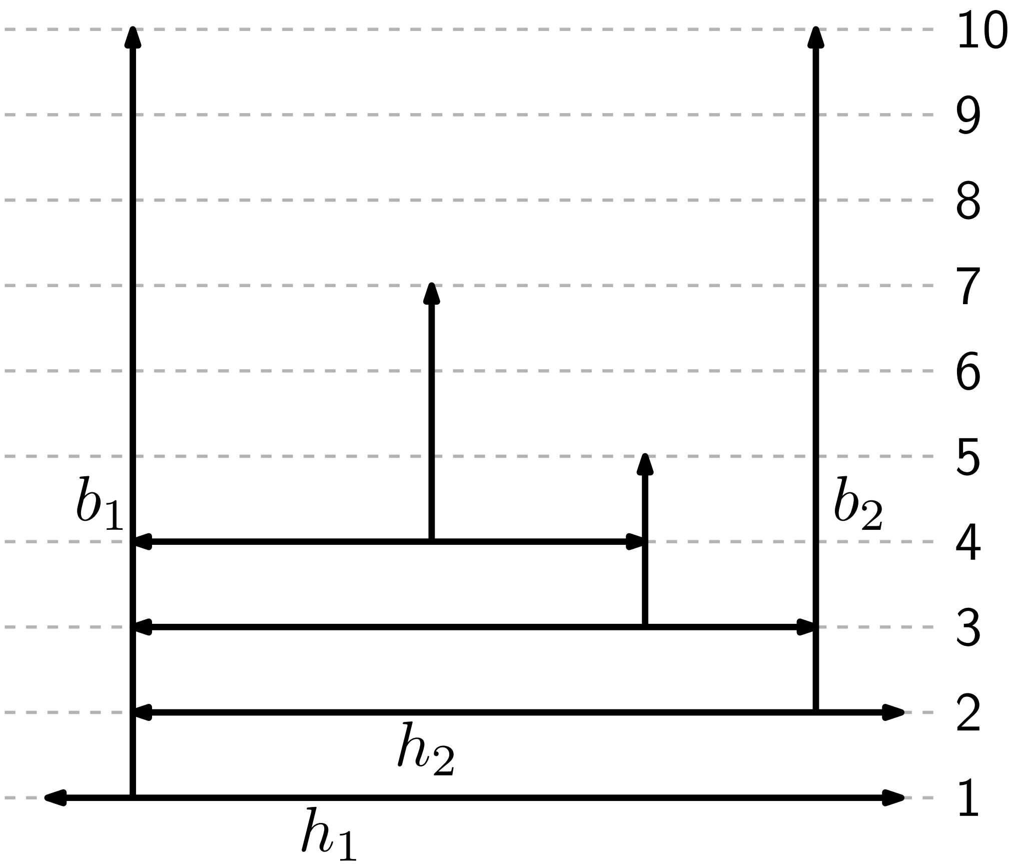




- betrachte Knoten in kanonischer Ordnung und zug. Schnyder Realizer
- $\Phi_i(k)$ sei Vater in T_i des Knotens v_k
- setze T's für v_1 und v_2 auf $y = 1$ und $y = 2$
- setze h_i auf $y = i$ mit linker und rechter Position von $\Phi_1(i)$ und $\Phi_2(i)$ und b_i von $y = i$ bis zum Index von $\Phi_3(i)$

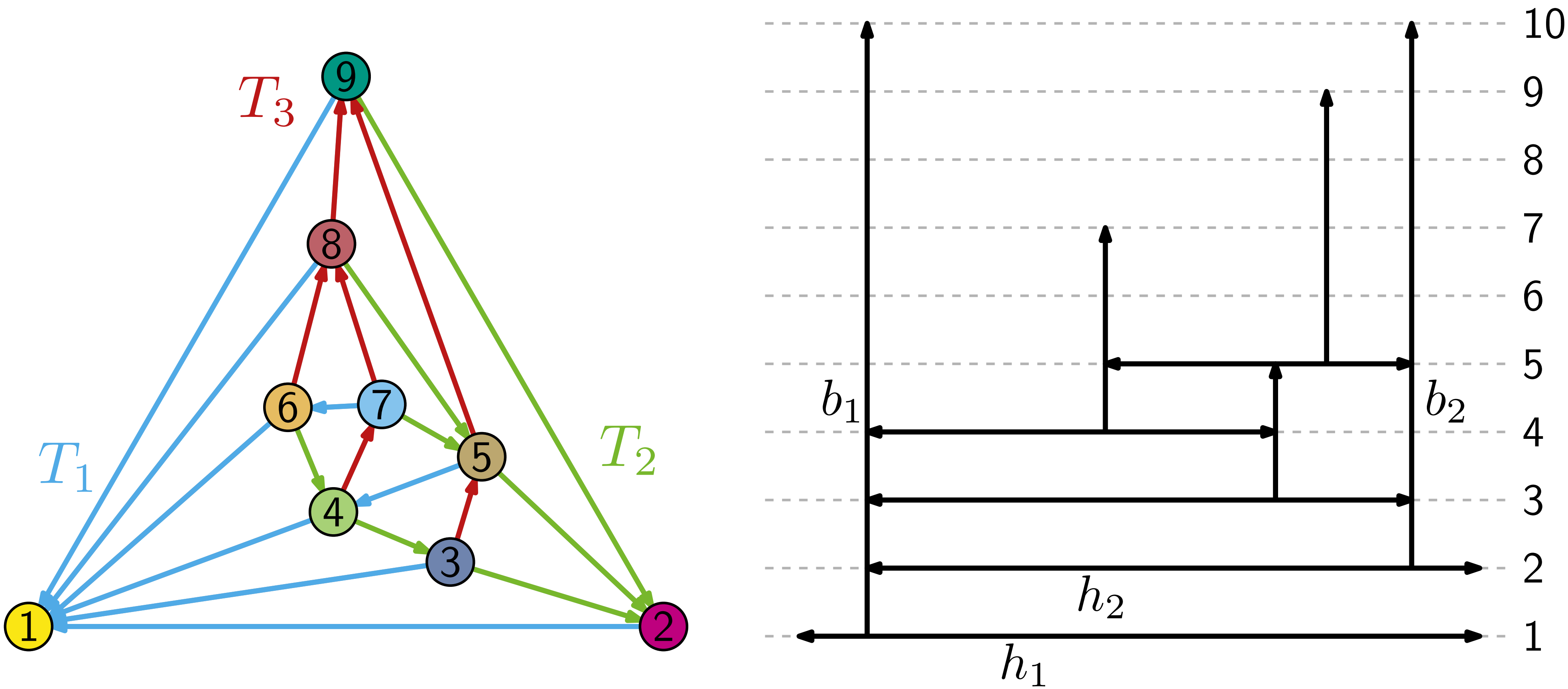
Phase 1: T-Kontaktrepräsentation





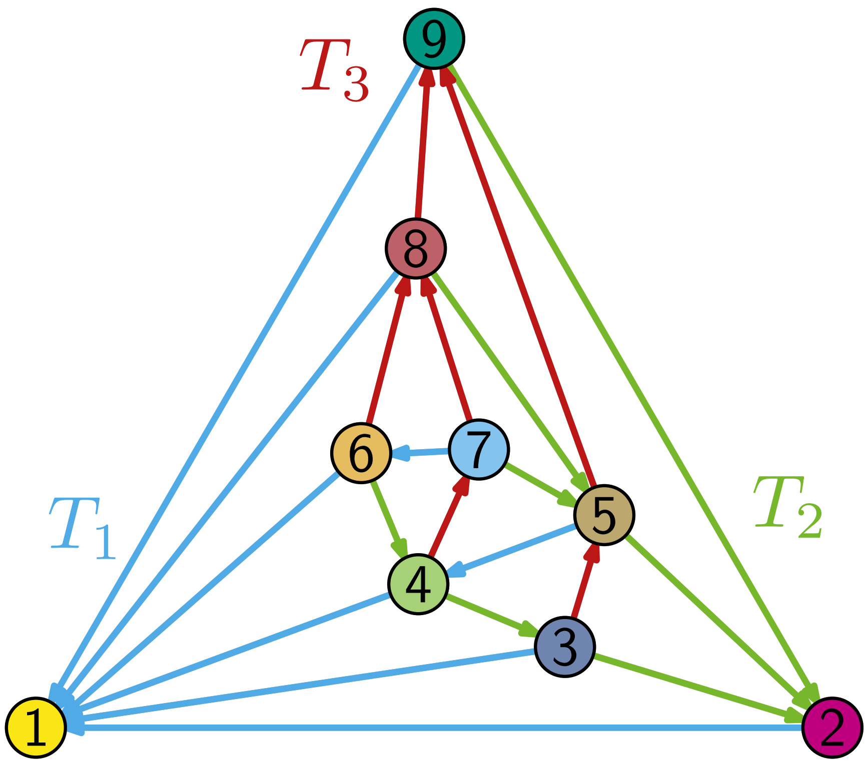
- betrachte Knoten in kanonischer Ordnung und zug. Schnyder Realizer
- $\Phi_i(k)$ sei Vater in T_i des Knotens v_k
- setze T's für v_1 und v_2 auf $y = 1$ und $y = 2$
- setze h_i auf $y = i$ mit linker und rechter Position von $\Phi_1(i)$ und $\Phi_2(i)$ und b_i von $y = i$ bis zum Index von $\Phi_3(i)$

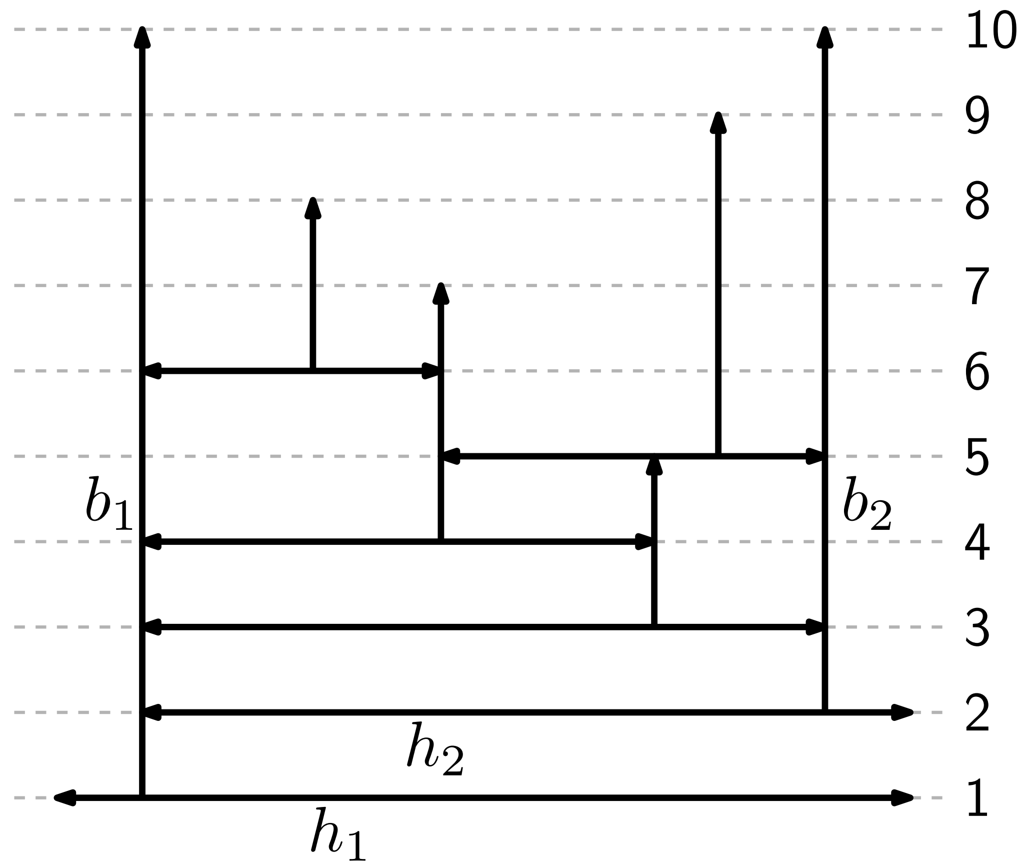
Phase 1: T-Kontaktrepräsentation



- betrachte Knoten in kanonischer Ordnung und zug. Schnyder Realizer
- $\Phi_i(k)$ sei Vater in T_i des Knotens v_k
- setze T's für v_1 und v_2 auf $y = 1$ und $y = 2$
- setze h_i auf $y = i$ mit linker und rechter Position von $\Phi_1(i)$ und $\Phi_2(i)$ und b_i von $y = i$ bis zum Index von $\Phi_3(i)$

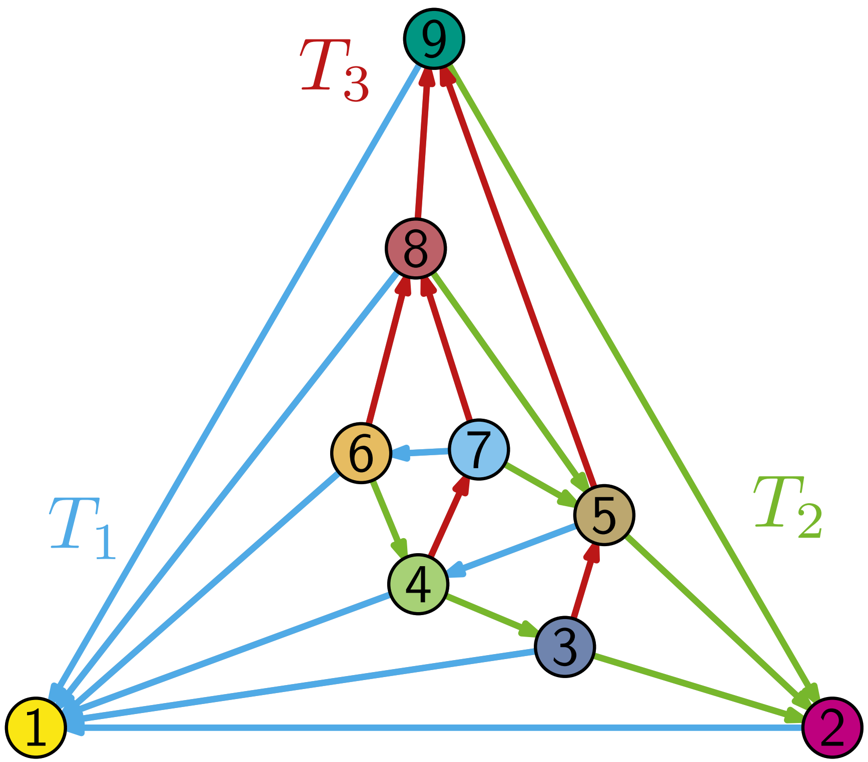
Phase 1: T-Kontaktrepräsentation

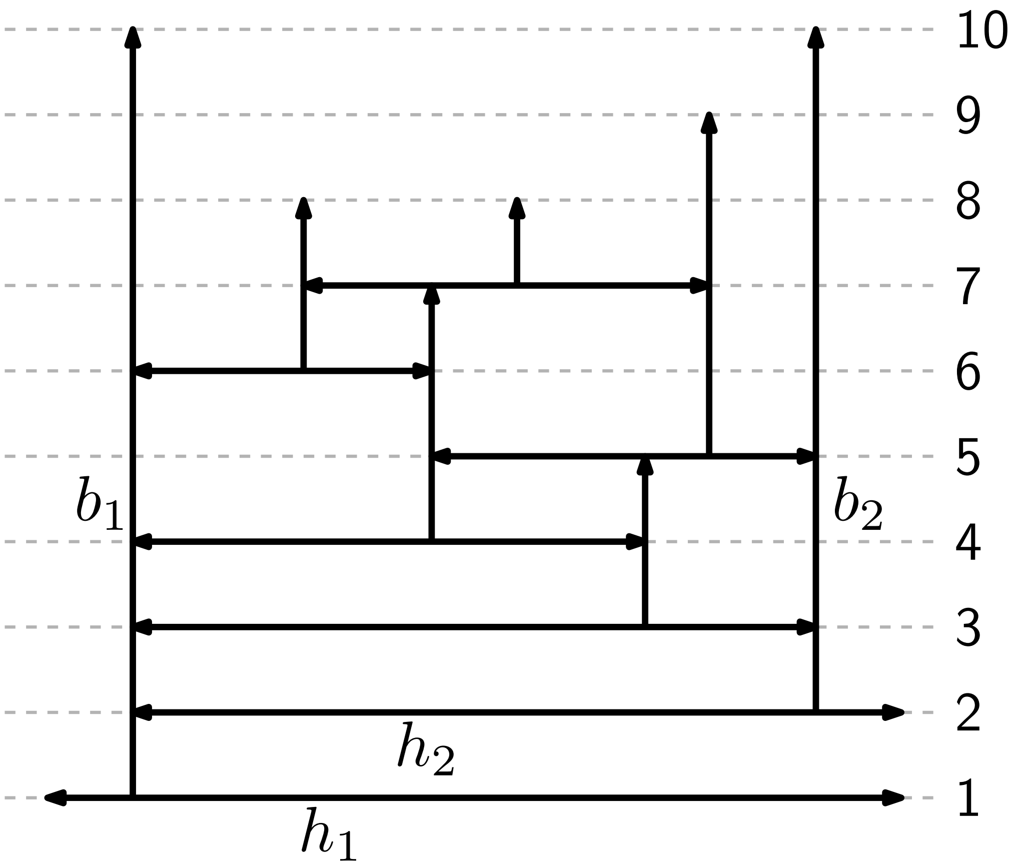




- betrachte Knoten in kanonischer Ordnung und zug. Schnyder Realizer
- $\Phi_i(k)$ sei Vater in T_i des Knotens v_k
- setze T's für v_1 und v_2 auf $y = 1$ und $y = 2$
- setze h_i auf $y = i$ mit linker und rechter Position von $\Phi_1(i)$ und $\Phi_2(i)$ und b_i von $y = i$ bis zum Index von $\Phi_3(i)$

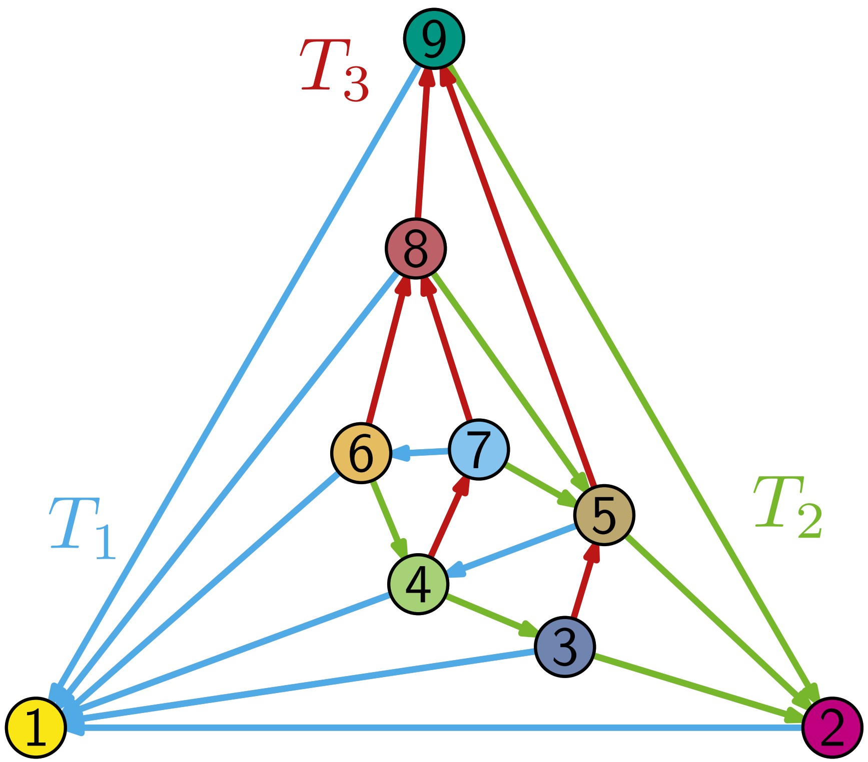
Phase 1: T-Kontaktrepräsentation

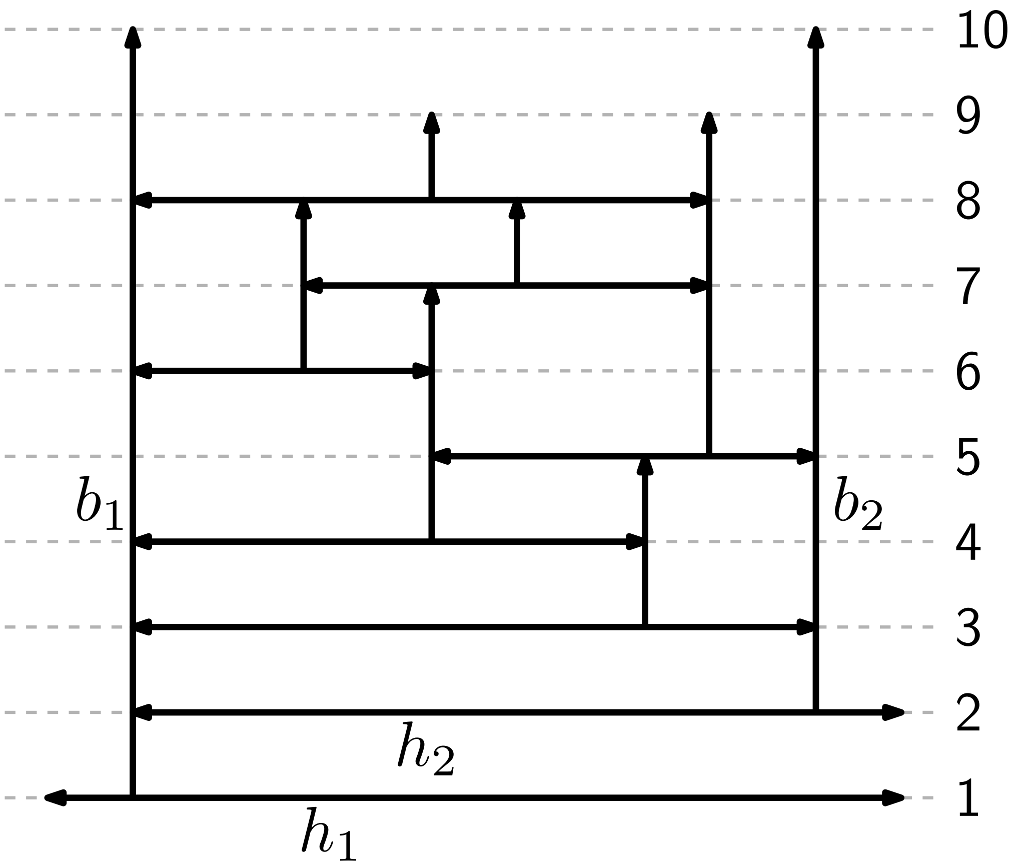




- betrachte Knoten in kanonischer Ordnung und zug. Schnyder Realizer
- $\Phi_i(k)$ sei Vater in T_i des Knotens v_k
- setze T's für v_1 und v_2 auf $y = 1$ und $y = 2$
- setze h_i auf $y = i$ mit linker und rechter Position von $\Phi_1(i)$ und $\Phi_2(i)$ und b_i von $y = i$ bis zum Index von $\Phi_3(i)$

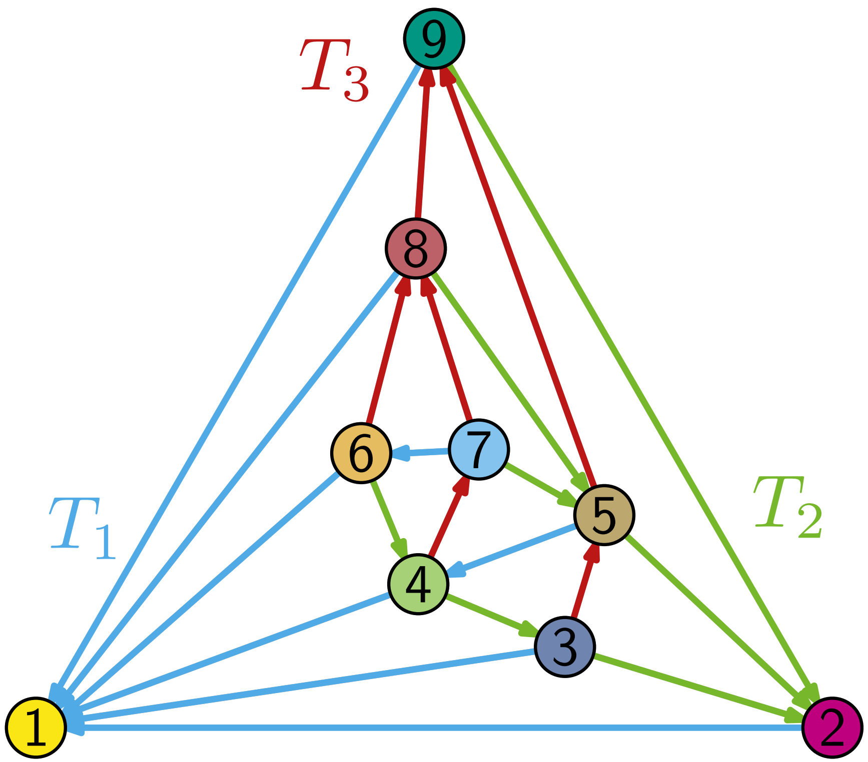
Phase 1: T-Kontaktrepräsentation

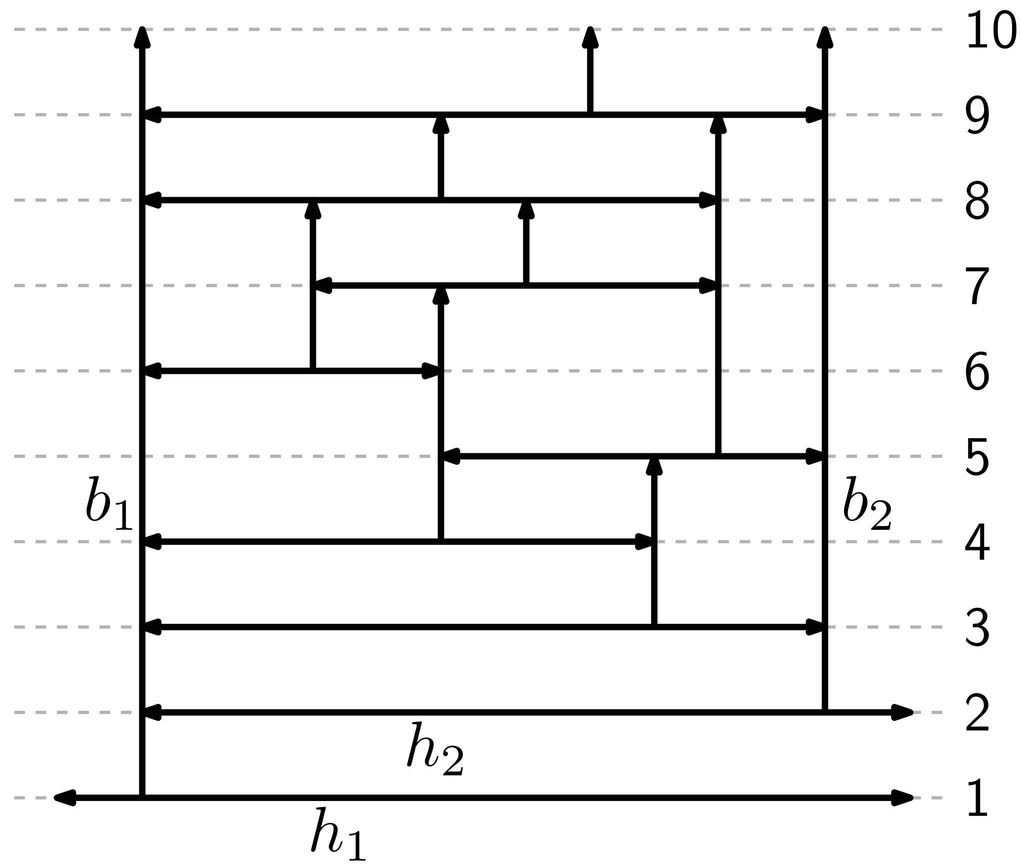




- betrachte Knoten in kanonischer Ordnung und zug. Schnyder Realizer
- $\Phi_i(k)$ sei Vater in T_i des Knotens v_k
- setze T's für v_1 und v_2 auf $y = 1$ und $y = 2$
- setze h_i auf $y = i$ mit linker und rechter Position von $\Phi_1(i)$ und $\Phi_2(i)$ und b_i von $y = i$ bis zum Index von $\Phi_3(i)$

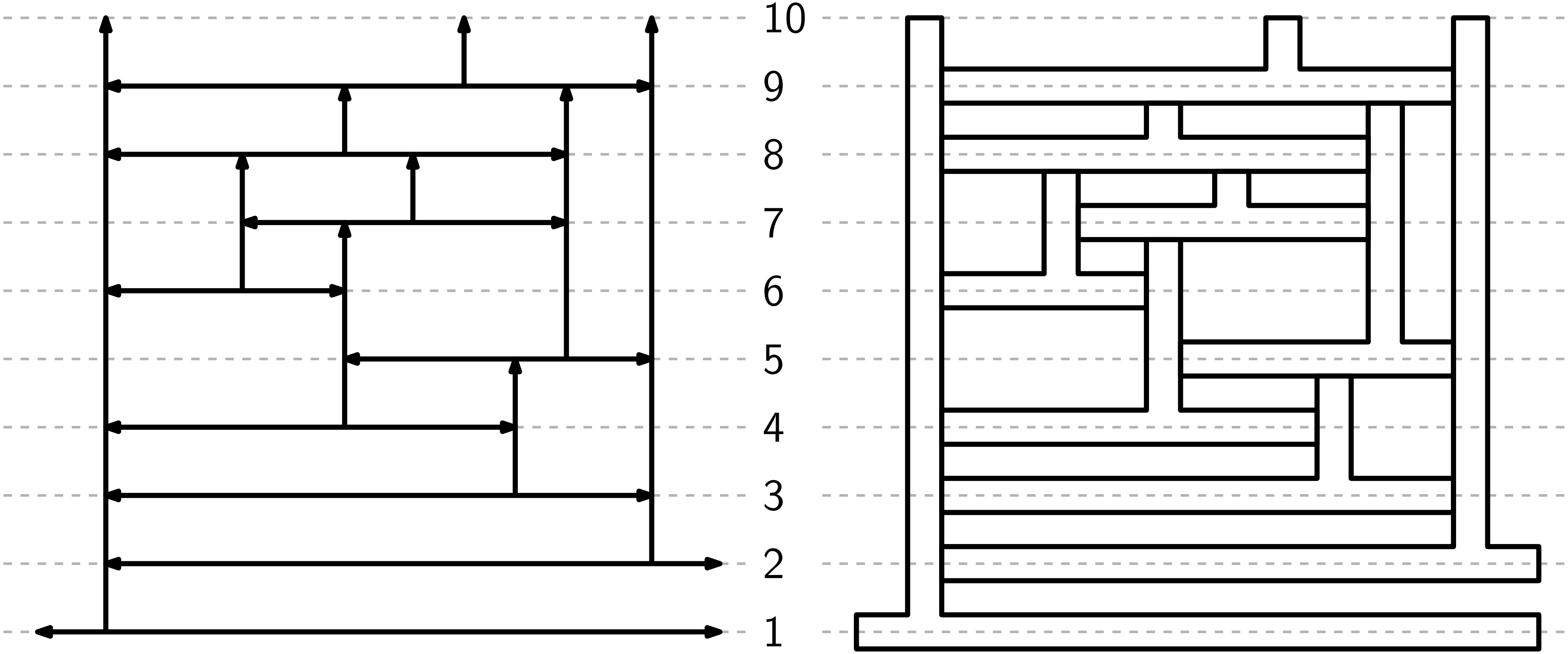
Phase 1: T-Kontaktrepräsentation





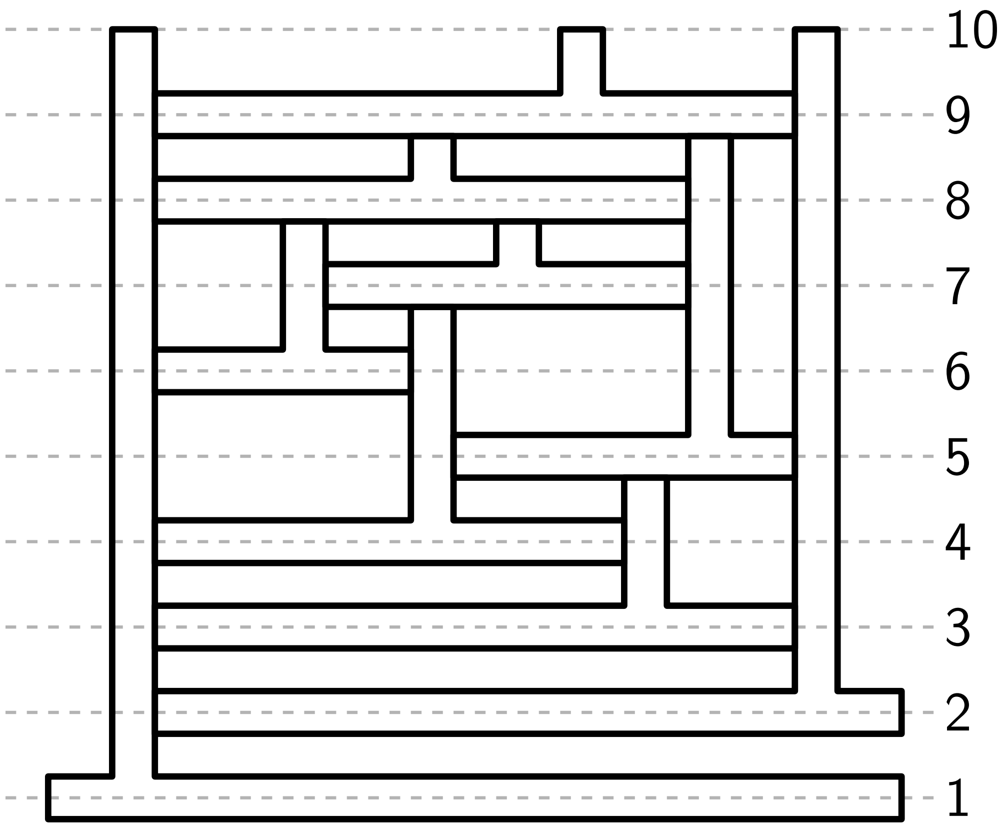
- betrachte Knoten in kanonischer Ordnung und zug. Schnyder Realizer
- $\Phi_i(k)$ sei Vater in T_i des Knotens v_k
- setze T's für v_1 und v_2 auf $y = 1$ und $y = 2$
- setze h_i auf $y = i$ mit linker und rechter Position von $\Phi_1(i)$ und $\Phi_2(i)$ und b_i von $y = i$ bis zum Index von $\Phi_3(i)$

Phase 2: λ-Verdickung

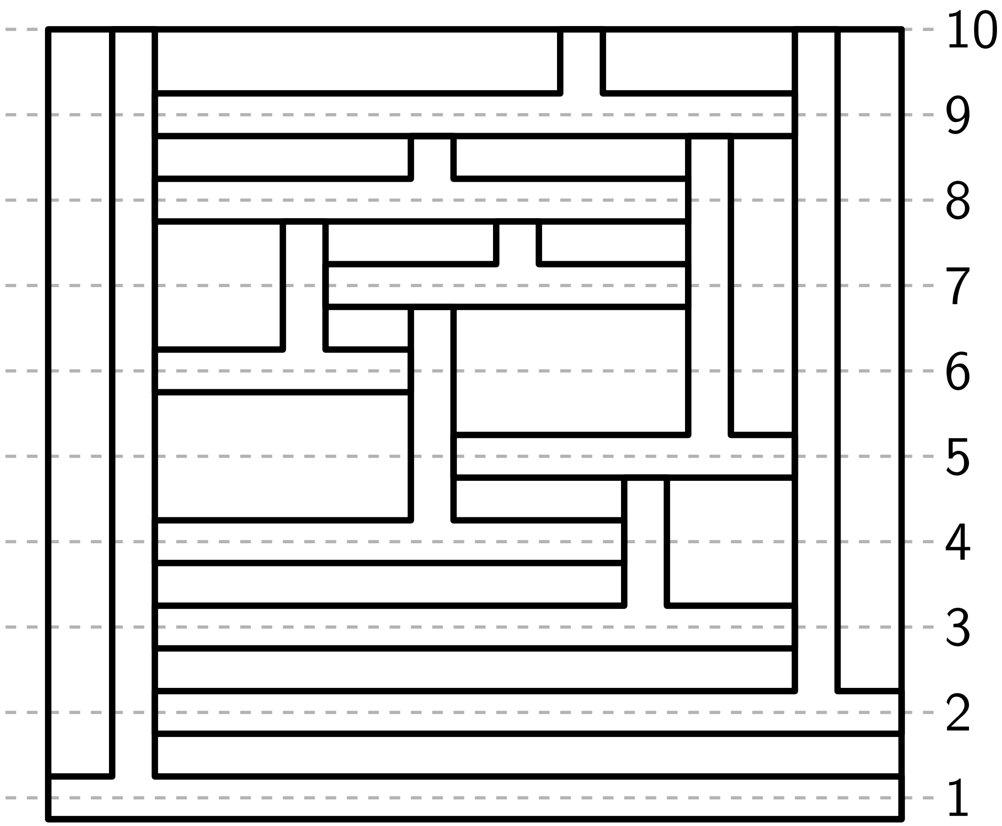


Ersetze jedes T durch ein T-förmiges Polygon mit Dicke λ für $0 < \lambda < 1/2$

Phase 3: Löcher entfernen

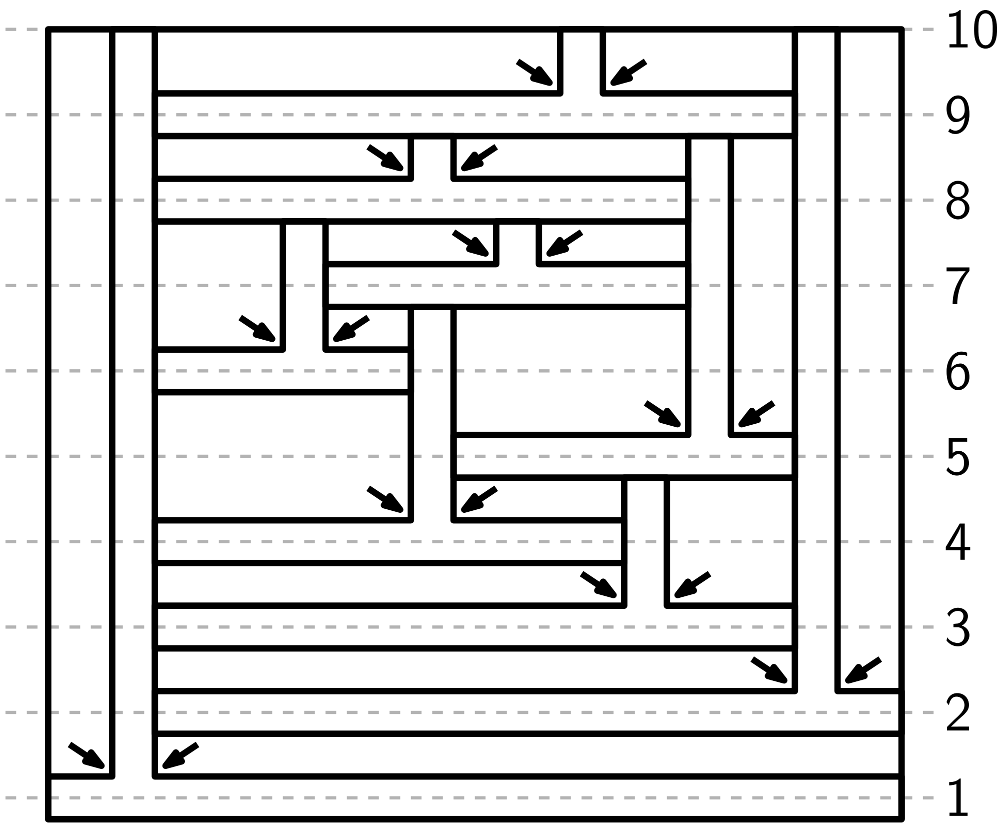


Phase 3: Löcher entfernen



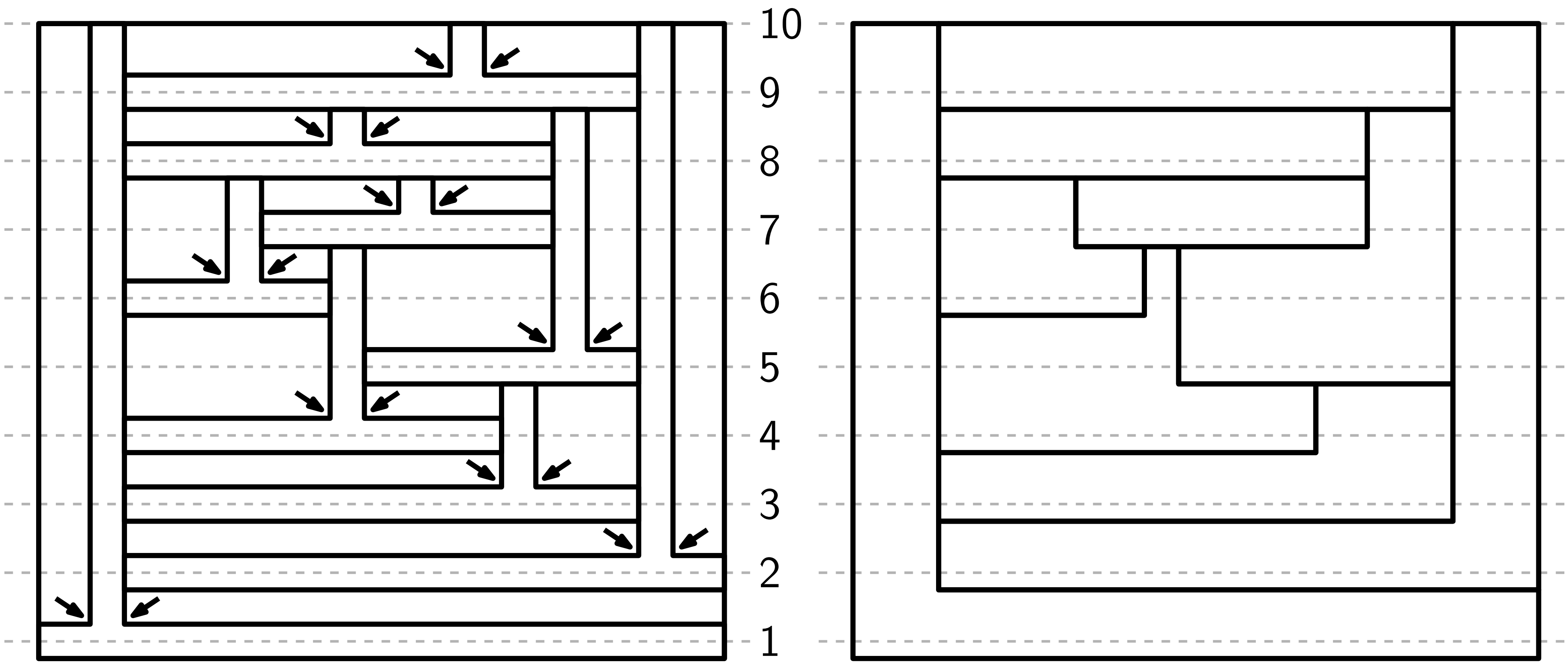
- setze Rahmen um die T-Polygone

Phase 3: Löcher entfernen



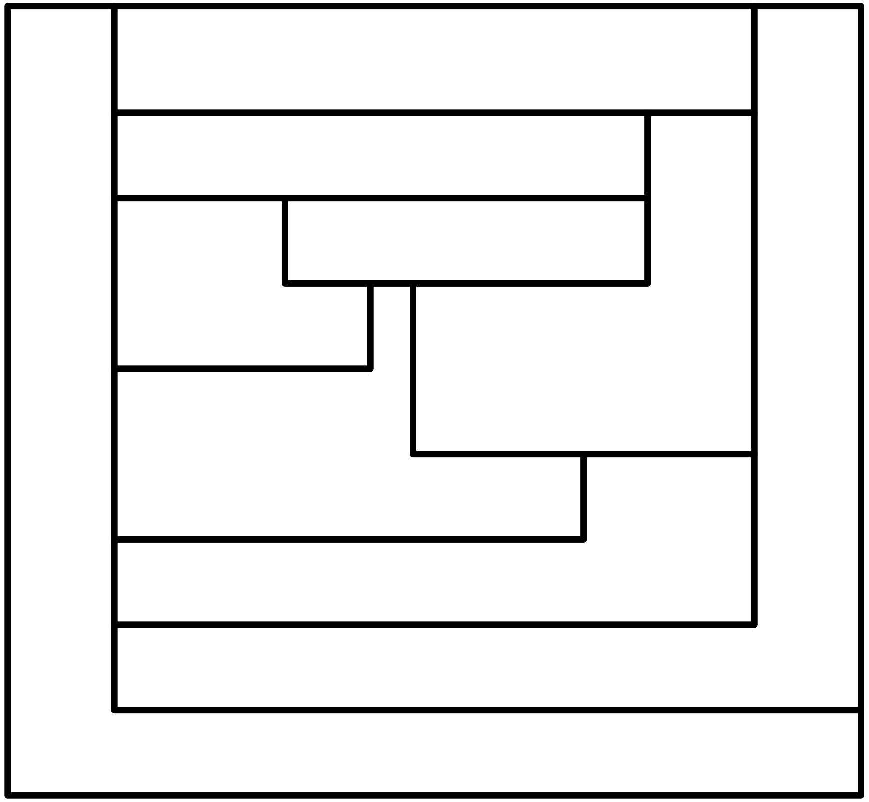
- setze Rahmen um die T-Polygone
- jedes (innere) Loch stammt von Dreiecks-Facette
- weise Löcher den eindeutigen konkaven Ecken zu

Phase 3: Löcher entfernen



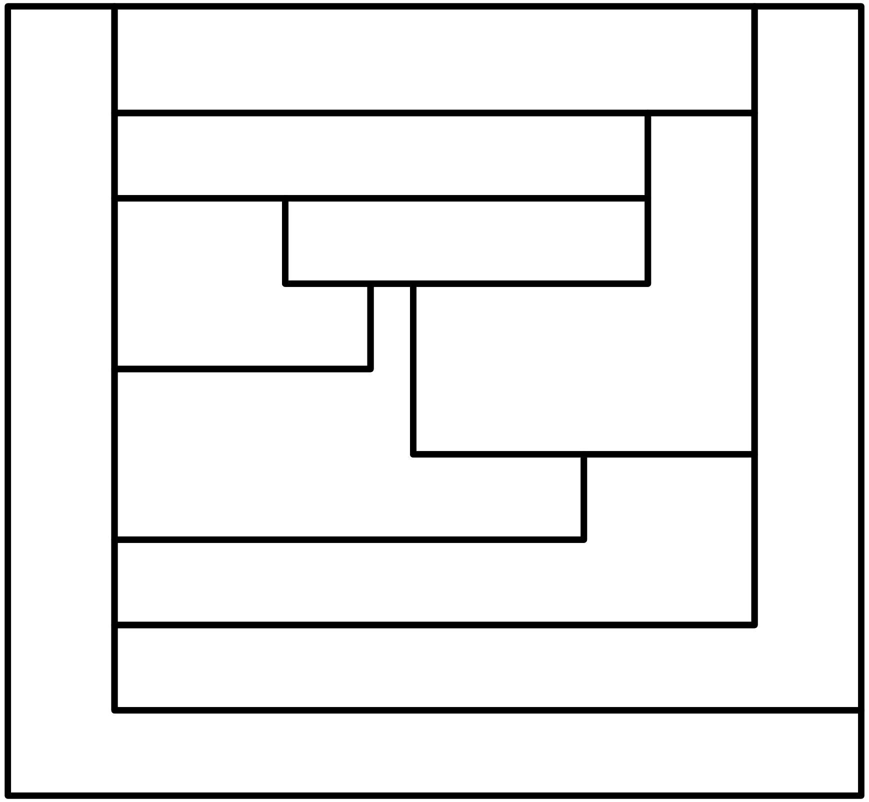
- setze Rahmen um die T-Polygone
- jedes (innere) Loch stammt von Dreiecks-Facette
- weise Löcher den eindeutigen konkaven Ecken zu
- es entsteht ein rektilineares Dual mit maximal 8-seitigen Polygonen

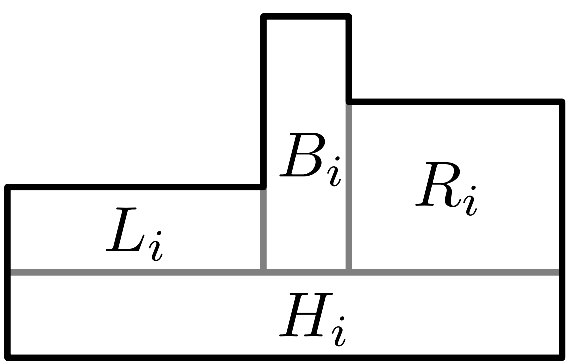
Flächenuniversalität



Satz:
Das erhaltene Layout ist flächenuniversell und nutzt nur Polygone mit Komplexität ≤ 8.

Flächenuniversalität



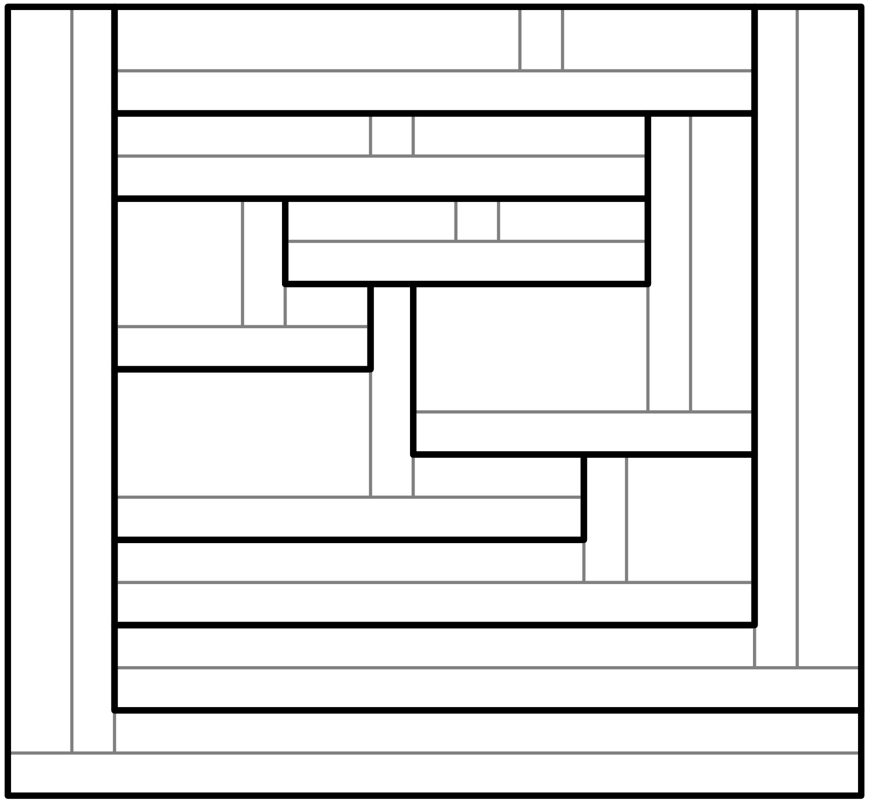


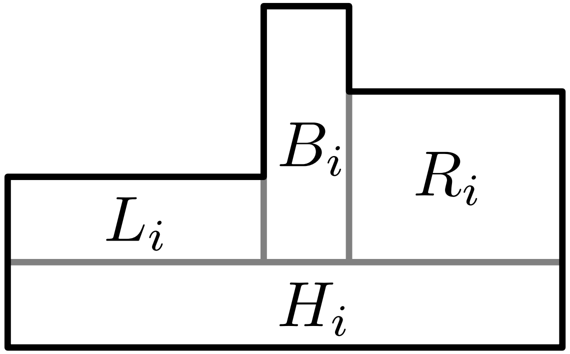
Satz:
Das erhaltene Layout ist flächenuniversell und nutzt nur Polygone mit Komplexität ≤ 8.

Beweis:

- zerteile jedes T-förmige Polygon in vier Rechtecke
- Rechteckslayout ist einseitig $\Rightarrow$ flächenuniversell
- teile Sollfläche für jede Region beliebig in vier Teile

Flächenuniversalität





Satz:
Das erhaltene Layout ist flächenuniversell und nutzt nur Polygone mit Komplexität ≤ 8.

Beweis:

- zerteile jedes T-förmige Polygon in vier Rechtecke
- Rechteckslayout ist einseitig $\Rightarrow$ flächenuniversell
- teile Sollfläche für jede Region beliebig in vier Teile

Algorithmische Kartografie in der Praxis

Karlsruhe Institute of Technology

Projekt zum Semesterabschluss (16.7.)

Welche Algorithmen werden in gängigen Kartografiesystemen tatsächlich verwendet?

Gruppe 1:
Online-Systeme

OpenStreetMap

Gruppe 2:
Professionelle GIS Software

esri

Gruppe 3:
Kartografie-Software

Avenza Systems Inc.
MAPublisher

Algorithmische Kartografie in der Praxis

Projekt zum Semesterabschluss (16.7.)

Welche Algorithmen werden in gängigen Kartografiesystemen tatsächlich verwendet?

Gruppe 1:
Online-Systeme

Gruppe 2:
Professionelle GIS Software

Gruppe 3:
Kartografie-Software

Finden Sie für Ihre Kategorie an Produkten heraus, welche Algorithmen für die in der Vorlesung behandelten Probleme z.B. zur Beschriftung, zur Linienvereinfachung, für Kartogramme implementiert sind.

Stellen Sie Ihre Ergebnisse in zwei Wochen in einer kurzen Präsentation von 10–15 Minuten vor.

Ablauf konkret

1) Gruppeneinteilung

Gruppe 1:	Gruppe 2:	Gruppe 3:
Online-Systeme	Professionelle GIS Software	Kartografie-Software

2) Interne Aufgabenverteilung und erste Recherchen

3) Treffen mit Betreuer (Freitag, 5.7.)

4) Detaillierte Untersuchung der Fragen, Erstellung der Präsentation

5) Treffen mit Betreuer (Freitag 12.7.)

6) Abschlusspräsentation der Ergebnisse (Dienstag 16.7.)